P517567.PDF [Page: 3 of 78]

Adaptive Systems for Signal processing

FINAL CONTRACTUAL REPORT

W7711-9-7589/001/TOR

By

Prof. Simon Haykin
McMaster University
Hamilton, Ontario, L8S 4L9

Contract Report: **CR 2002-045**

Scientific Authority: **Dr. Stergios Stergiopoulos**
DRDC-Toronto
1133 Sheppard Ave. West
Toronto, Ontario, M3M 3B9

March 2002

P517567.PDF [Page: 4 of 78]

104-05626/

Abstract

The present report discusses in some detail the recent developments in adaptive signal processing and their advantages for real-time system applications. It reviews the basic issues involved in the study of adaptive systems for signal processing. The virtues of this approach to statistical signal processing may be summarized as follows:

- The use of an adaptive filtering algorithm, which enables the system to adjust its free parameters (in a supervised or unsupervised manner) in accordance with the underlying statistics of the environment in which the system operates. Hence, the need for determining the statistical characteristics of the environment is avoided.
- Tracking capability, which permits the system to follow statistical variations (i.e. non-stationarity) of the environment.
- The availability of many different adaptive filtering algorithms, both linear and non-linear, which can be used to deal with a wide variety of signal-processing applications in radar, sonar and biomedical imaging.
- Digital implementation of the adaptive filtering algorithms, which can be carried out in hardware or software form.

In many cases these advanced algorithms trade robustness for improved performance. Furthermore, the improvements achieved are generally not uniform across all signal and noise environments of operational scenarios. The challenge is to develop a concept, which allows an appropriate mixture of these algorithms to be implemented in practical real time systems.

In general, improvements in system performance or array gain improvements are required when the noise environment of an operational system is non-isotropic, such as the noise environment of: (1) atmospheric noise or clutter (radar applications), (2) cluttered coastal waters and areas with high shipping density in which sonar systems operate (sonar applications), and the complexity of the human body (medical imaging applications). An alternative approach to improve the array gain of a real-time system requires the deployment of very large aperture arrays, which leads to technical and operational implications. Thus, the implementation of non-conventional signal processing schemes in operational systems will minimize very costly hardware requirements associated with array gain improvements.

P517567.PDF [Page: 5 of 78]

Adaptive Systems for Signal Processing†
Simon Haykin

1. The Filtering Problem

The term filter is often used to describe a device in the form of a piece of physical hardware or software that is applied to a set of noisy data in order to extract information about a prescribed quantity of interest. The noise may arise from a variety of sources. For example, the data may have been derived by means of noisy sensors or may represent a useful signal component that has been corrupted by transmission through a communication channel. In any event, we may use a filter to perform three basic information-processing tasks:

1. Filtering, which means the extraction of information about a quantity of interest at time t by using data measured up to and including time t.
2. Smoothing, which differs from filtering in that information about the quantity of interest need not be available at time t, and data measured later than time t can be used in obtaining this information. This means that in the case of smoothing there is a delay in producing the result of interest. Since in the smoothing process we are able to use data obtained not only up to time t but also data obtained after time t, we would expect smoothing to be more accurate in some sense than filtering.
3. Prediction, which is the forecasting side of information processing. The aim here is to derive information about what the quantity of interest will be like at some time $t + \tau$ in the future, for some $\tau > 0$, by using data measured up to and including time t.

We may classify filters into linear and nonlinear. A filter is said to be linear if the filtered, smoothed, or predicted quantity at the output of the device is a linear function of the observations applied to the filter input. Otherwise, the filter is nonlinear.

In the statistical approach to the solution of the linear filtering problem as classified above, we assume the availability of certain statistical parameters (i.e., mean and correlation functions) of the useful signal and unwanted additive noise, and the requirement is to design a linear filter with the noisy data as input so as to minimize the effects of noise at the filter output according to some statistical criterion. A useful approach to this filter-optimization problem is to minimize the mean-square value of the error signal that is defined as the difference between some desired response and the actual filter output. For stationary inputs, the resulting solution is commonly known as the Wiener filter, which is said to be optimum in the mean-square sense. A plot of the mean-square value of the error signal versus the adjustable parameters of a linear filter is referred to as the error-performance surface. The minimum point of this surface represents the Wiener solution.

The Wiener filter is inadequate for dealing with situations in which nonstationarity of the signal and/or noise is intrinsic to the problem. In such situations, the optimum filter has to assume a

---

†The material presented in this chapter is based on the author's two textbooks: (1) Adaptive Filter Theory (1996), and (2) Neural Networks: A Comprehensive Foundation, Prentice-Hall (1999).

517567.PDF [Page: 6 of 78]

time-varying form. A highly successful solution to this more difficult problem is found in the Kalman filter, a powerful device with a wide variety of engineering applications.

Linear filter theory, encompassing both Wiener and Kalman filters, has been developed fully in the literature for continuous-time as well as discrete-time signals. However, for technical reasons influenced by the wide availability of digital computers and the ever-increasing use of digital signal-processing devices, we find in practice that the discrete-time representation is often the preferred method. Accordingly, in this chapter, we only consider the discrete-time version of Wiener and Kalman filters. In this method of representation, the input and output signals, as well as the characteristics of the filters themselves, are all defined at discrete instants of time. In any case, a continuous-time signal may always be represented by a sequence of samples that are derived by observing the signal at uniformly spaced instants of time. No loss of information is incurred during this conversion process provided, of course, we satisfy the well-known sampling theorem, according to which the sampling rate has to be greater than twice the highest frequency component of the continuous-time signal (assumed to be of a low-pass kind). We may thus represent a continuous-time signal $u(t)$ by the sequence $u(n), n = 0, \pm 1, = \pm 2, \cdots$, where for convenience we have normalized the sampling period to unity, a practice that we follow throughout this chapter.

2. Adaptive Filters

The design of a Wiener filter requires a priori information about the statistics of the data to be processed. The filter is optimum only when the statistical characteristics of the input data match the a priori information on which the design of the filter is based. When this information is not known completely, however, it may not be possible to design the Wiener filter or else the design may no longer be optimum. A straightforward approach that we may use in such situations is the "estimate and plug" procedure. This is a two-stage process whereby the filter first "estimates" the statistical parameters of the relevant signals and then "plugs" the results so obtained into a nonrecursive formula for computing the filter parameters. For real-time operation, this procedure has the disadvantage of requiring excessively elaborate and costly hardware. A more efficient method is to use an adaptive filter. By such a device we mean one that is self-designing in that the adaptive filter relies for its operation on a recursive algorithm, which makes it possible for the filter to perform satisfactorily in an environment where complete knowledge of the relevant signal characteristics is not available. The algorithm starts from some predetermined set of initial conditions, representing whatever we know about the environment. Yet, in a stationary environment, we find that after successive iterations of the algorithm it converges to the optimum Wiener solution in some statistical sense. In a nonstationary environment, the algorithm offers a tracking capability, in that it can track time variations in the statistics of the input data, provided that the variations are sufficiently slow.

As a direct consequence of the application of a recursive algorithm whereby the parameters of an adaptive filter are updated from one iteration to the next, the parameters become data dependent. This, therefore, means that an adaptive filter is in reality a nonlinear device, in the sense that it does not obey the principle of superposition. Notwithstanding this property, adaptive filters are commonly classified as linear or nonlinear. An adaptive filter is said to be linear if the estimate of quantity of interest is computed adaptively (at the output of the filter) as a linear combination of

P517567.PDF [Page: 7 of 78]

the available set of observations applied to the filter input. Otherwise the adaptive filter is said to be nonlinear.

A wide variety of recursive algorithms have been developed in the literature of the operation of linear adaptive filters. In the final analysis, the choice of one algorithm over another is determined by one or more of the following factors:

- Rate of convergence. This is defined as the number of iterations required for the algorithm, in response to stationary inputs, to converge "close enough" to the optimum Wiener solution in the mean-square sense. A fast rate of convergence allows the algorithm to adapt rapidly to a stationary environment of unknown statistics.
- Misadjustment. For an algorithm of interest, this parameter provides a quantitative measure of the amount by which the final value of the mean-squared error, averaged over an ensemble of adaptive filters, deviates from the minimum mean-squared error that is produced by the Wiener filter.
- Tracking. When an adaptive filtering algorithm operates in a nonstationary environment, the algorithm is required to track statistical variations in the environment. The tracking performance of the algorithm, however, is influenced by two contradictory features: (1) rate of convergence, and (b) steady-state fluctuation due to algorithm noise.
- Robustness. For an adaptive filter to be robust, small disturbances (i.e., disturbances with small energy) can only result in small estimation errors. The disturbances may arise from a variety of factors, internal or external to the filter.
- Computational requirements. Here the issues of concern include (a) the number of operations (i.e., multiplications, divisions, and additions/subtractions) required to make one complete iteration of the algorithm, (b) the size of memory locations required to store the data and the program, and (c) the investment required to program the algorithm on a computer.
- Structure. This refers to the structure of information flow in the algorithm, determining the manner in which it is implemented in hardware form. For example, an algorithm whose structure exhibits high modularity, parallelism, or concurrency is well suited for implementation using very large-scale integration (VLSI).[1]
- Numerical properties. When an algorithm is implemented numerically, inaccuracies are produced due to quantization errors. The quantization errors are due to analog-to-digital conversion of the input data and digital representation of internal calculations. Ordinarily, it is the latter source of quantization errors that poses a serious design problem. In particular, there are two basic issues of concern: numerical stability and numerical accuracy. Numerical stability is an inherent characteristic of an adaptive filtering algorithm. Numerical accuracy, on the other hand, is determined by the number of bits (i.e., binary digits used in the numerical representation of data samples and filter coefficients). An adaptive filtering algorithm is said to

---

[1] VLSI technology favors the implementation of algorithms that possess high modularity, parallelism, or concurrency. We say that a structure is modular when it consists of similar stages connected in cascade. By parallelism we mean a large number of operations being performed side by side. By concurrency we mean a large number of similar computations being performed at the same time.

For a discussion of VLSI implementation of adaptive filters, see Shanbhag and Parhi (1994). This book emphasizes the use of pipelining, an architectural technique used for increasing the throughput of an adaptive filtering algorithm.

17567.PDF [Page: 8 of 78]

be numerically robust when it is insensitive to variations in the wordlength used in its digital implementation.

These factors, in their own ways, also enter into the design of nonlinear adaptive filters, except for the fact that we now no longer have a well-defined frame of reference in the form of a Wiener filter. Rather, we speak of a nonlinear filtering algorithm that may converge to a local minimum or, hopefully, a global minimum on the error-performance surface.

In the sections that follow, we shall first discuss various aspects of linear adaptive filters. Discussion of nonlinear adaptive filters is deferred to a later section in the chapter.

3. Linear Filter Structures

The operation of a linear adaptive filtering algorithm involves two basic processes: (1) a filtering process designed to produce an output in response to a sequence of input data, and (2) an adaptive process, the purpose of which is to provide a mechanism for the adaptive control of an adjustable set of parameters used in the filtering process. These two processes work interactively with each other. Naturally, the choice of a structure for the filtering process has a profound effect on the operation of the algorithm as a whole.

There are three types of filter structures that distinguish themselves in the context of an adaptive filter with finite memory or, equivalently, finite-duration impulse response. The three filter structures are as follows:

1. Transversal filter. The transversal filter,[2] also referred to as a tapped-delay line filter, consists of three basic elements, as depicted in Fig. 1: (a) unit-delay element, (b) multiplier, and (c) adder. The number of delay elements used in the filter determines the finite duration of its impulse response. The number of delay elements, shown as M-1 in Fig. 1, is commonly referred to as the filter order. In this figure, the delay elements are each identified by the unit-delay operator z^{-1}. In particular, when z^{-1} operates on the input $u(n)$, the resulting output is $u(n\text{-}1)$. The role of each multiplier in the filter is to multiply the tap input, to which it is connected by a filter coefficient referred to as a tap weight. Thus a multiplier connected to the kth tap input $u(n\text{-}k)$ produces the scalar version of the inner product, $w_k^* u(n-k)$, where w_k is the respective tap weight and $k = 0, 1, \cdots, M-1$. The asterisk denotes complex conjugation, which assumes that the tap inputs and therefore the tap weights are all complex valued. The combined role of the adders in the filter is to sum the individual multiplier outputs and produce an overall filter output. For the transversal filter described in Fig. 1, the filter output is given by

---

[2] The transversal filter was first described by Kallmann as a continuous-time device whose output is formed as a linear combination of voltages taken from uniformly spaced taps in a nondispersive delay line (Kallmann, 1940). In recent years, the transversal filter has been implemented using digital circuitry, charge-coupled devices, or surface-acoustic wave devices. Owing to its versatility and ease of implementation, the transversal filter has emerged as an essential signal-processing structure in a wide variety of applications.

P517567.PDF [Page 9 of 78]

$$y(n) = \sum_{k=0}^{m-1} w_k^* u(n-k) \tag{1}$$

Equation (1) is called a finite convolution sum in the sense that it convolves the finite-duration impulse response of the filter, w_n^*, with the filter input $u(n)$ to produce the filter output $y(n)$.

2. Lattice predictor. A lattice predictor[3] is modular in structure in that it consists of a number of individual stages, each of which has the appearance of a lattice, hence, the name "lattice" as a structural descriptor. Figure 2 depicts a lattice predictor consisting of M-1 stages; the number M-1 is referred to as the predictor order. The mth stage of the lattice predictor in Fig. 2 is described by the pair of input-output relations (assuming the use of complex-valued, wide-sense stationary input date):

$$f_m(n) = f_{m-1}(n) + \kappa_m^* b_{m-1}(n-1) \tag{2}$$

$$b_m(n) = b_{m-1}(n-1) + \kappa_m f_{m-1}(n) \tag{3}$$

where $m = 1, 2, \cdots, M-1$, and M-1 is the final predictor order. The variable $f_m(n)$ is the mth forward prediction error, and $b_m(n)$ is the mth backward prediction error. The coefficient κ_m is called the mth reflection coefficient. The forward prediction error $f_m(n)$ is defined as the difference between the input $u(n)$ and its one-step predicted value; the latter is based on the set of m past inputs $u(n-1), \cdots, u(n-m)$. Correspondingly, the backward prediction error $b_m(n)$ is defined as the difference between the input $u(n-m)$ and its "backward" prediction based on the set of m "future" inputs $u(n), \cdots, u(n-m+1)$. Considering the conditions at the input of stage 1 in Fig. 2, we have

$$f_0(n) = b_o(n) = u(n) \tag{4}$$

where $u(n)$ is the lattice predictor input at time n. Thus, starting with the initial conditions of Eq. (4) and given the set of reflection coefficients $\kappa_1, \kappa_2, \cdots, \kappa_{M-1}$, we may determine the final pair of outputs $f_{M-1}(n)$ and $b_{M-1}(n)$ by moving through the lattice predictor, stage by stage.

For a correlated input sequence $u(n), u(n-1), \cdots, u(n-M+1)$ drawn from a stationary process, the backward prediction errors $b_0, b_1(n), \cdots, b_{M-1}(n)$ form a sequence of uncorrelated random variables. Moreover, there is a one-to-one correspondence between these two sequences of random variables in the sense that if we are given one of them, we may uniquely determine the other, and vice versa. Accordingly, a linear combination of the backward prediction errors $b_0, b_1(n), \cdots, b_{M-1}(n)$ may be used to provide an estimate of some desired response $d(n)$, as depicted in the lower half of Fig. 2. The arithmetic difference between $d(n)$ and the estimate so produced represents the estimation error $e(n)$. The process described herein is referred to as a joint-process estimation. Naturally, we may use the original input sequence

[3] The development of the lattice predictor is credited to Itakura and Saito (1972).

517567.PDF [Page: 10 of 78]

$u(n), u(n-1), \cdots, u(n-M+1)$ to produce an estimate of the desired response $d(n)$ directly. The indirect method depicted in Fig. 2, however, has the advantage of simplifying the computation of the tap weights $h_0, h_1(n), \cdots, h_{M-1}$ by exploiting the uncorrelated nature of the corresponding backward prediction errors used in the estimation.

3. Systolic array. A systolic array[4] represents a parallel computing network ideally suited for mapping a number of important linear algebra computations, such as matrix multiplication, triangularization, and back substitution. Two basic types of processing elements may be distinguished in a systolic array: boundary cells and internal cells. Their functions are depicted in Figs. 3(a) and 3(b), respectively. In each case, the parameter r represents a value stored within the cell. The function of the boundary cell is to produce an output equal to the input u divided by the number r stored in the cell. The function of the internal cell is twofold: (a) to multiply the input z (coming in from the top) by the number r stored in the cell, subtract the product rz from the second input (coming in from the left), and thereby produce the difference $u - rz$ as an output from the right-hand side of the cell, and (b) to transmit the first z downward without alteration.

Consider, for example, the 3-by-3 triangular array shown in Fig. 4. This systolic array involves a combination of boundary and internal cells. In this case, the triangular array computes an output vector **y** related to the input vector **u** as follows:

$$\mathbf{y} = \mathbf{R}^{-T}\mathbf{u} \tag{5}$$

where the $\mathbf{R}^{-T}$ is the inverse of the transposed matrix $\mathbf{R}^{T}$. The elements of $\mathbf{R}^{T}$ are the respective cell contents of the triangular array. The zeros added to the inputs of the array in Fig. 4 are intended to provide the delays necessary for pipelining the computation described in Eq. 5.

A systolic array architecture, as described herein, offers the desirable features of modularity, local interconnections, and highly pipelined and synchronized parallel processing; the synchronization is achieved by means of a global clock.

We note that the transversal filter of Fig. 1, the joint-process estimator of Fig. 2 based on a lattice predictor, and the triangular systolic array of Fig. 4 have a common property: all three of them are characterized by an impulse response of finite duration. In other words, they are examples of a finite-duration impulse response (FIR) filter, whose structures contain feedforward paths only. On the other hand, the filter structure shown in Fig. 5 is an example of an infinite-duration impulse response (IIR) filter. The feature that distinguishes an IIR filter from an FIR filter is the inclusion of feedback paths. Indeed, it is the presence of feedback that makes the duration of the impulse response of an IIR filter infinitely long. Furthermore, the presence of feedback introduces a new problem, namely, that of stability. In particular, it is possible for an IIR filter to become unstable (i.e., break into oscillation), unless special precaution is taken in the choice of feedback

[4] The systolic array was pioneered by Kung and Leiserson (1978). In particular, the use of systolic arrays has made it possible to achieve a high throughput, which is required for many advanced signal processing algorithms to operate in real time.

P517567.PDF [Page: 11 of 78]

coefficients. By contrast, an FIR filter in inherently stable. This explains the reason for the popular use of FIR filters, in one form or another, as the structural basis for the design of linear adaptive filters.

4. Approaches to the Development of Linear Adaptive Filtering Algorithms

There is no unique solution to the linear adaptive filtering problem. Rather, we have a "kit of tools" represented by a variety of recursive algorithms, each of which offers desirable features of its own. (For complete detailed treatment of linear adaptive filters, see the book by Haykin (1996).) The challenge facing the user of adaptive filtering is, first, to understand the capabilities and limitations of various adaptive filtering algorithms and, second, to use this understanding in the selection of the appropriate algorithm for the application at hand.

Basically, we may identify two distinct approaches for deriving recursive algorithms for the operation of linear adaptive filters, as discussed next.

Stochastic Gradient approach

Here we may use a tapped-delay line or transversal filter as the structural basis for implementing the linear adaptive filter. For the case of stationary inputs, the cost function,[5] also referred to as the index of performance, is defined as the mean-squared error (i.e., the mean-square value of the difference between the desired response and the transversal filter output). This cost function is precisely a second-order function of the tap weights in the transversal filter. The dependence of the mean-squared error on the unknown tap weights may be viewed to be in the form of a multidimensional paraboloid (i.e., punch bowl) with a uniquely defined bottom or minimum point. as mentioned previously, we refer to this paraboloid as the error-performance surface; the tap weights corresponding to the minimum point of the surface define the optimum Wiener solution.

To develop a recursive algorithm for updating the tap weights of the adaptive transversal filter, we proceed in two stages. We first modify the system of Wiener-Hopf equations (i.e., the matrix equation defining the optimum Wiener solution) through the use of the method of steepest descent, a well-known technique in optimization theory. This modification requires the use of a gradient vector, the value of which depends on two parameters: the correlation matrix of the tap inputs in the transversal filter, and the cross-correlation vector between the desired response and the same tap inputs. Next, we use instantaneous values for these correlations so as to derive an estimate for the gradient vector, making it assume a stochastic character in general. The resulting algorithm is widely known as the least-mean-square (LMS) algorithm, the essence of which may be described in words as follows for the case of a transversal filter operating on real-valued data:

---

[5] In the general definition of a function, we speak of a transformation from a vector space into the space of real (or complex) scalars (Luenberger, 1969; Dorny, 1975). A cost function provides a quantitative measure for assessing the quality of performance; hence the restriction of it to a real scalar.

517567.PDF [Page: 12 of 78]

$$\begin{pmatrix}\text{updated value}\\ \text{of tap-weight}\\ \text{vector}\end{pmatrix} = \begin{pmatrix}\text{old value}\\ \text{of tap-weight}\\ \text{vector}\end{pmatrix} + \begin{pmatrix}\text{learning-}\\ \text{rate}\\ \text{parameter}\end{pmatrix} \begin{pmatrix}\text{tap-}\\ \text{input}\\ \text{vector}\end{pmatrix} \begin{pmatrix}\text{error}\\ \text{signal}\end{pmatrix}$$

where the error signal is defined as the difference between some desired response and the actual response of the transversal filter produced by the tap-input vector.

The LMS algorithm, summarized in Table 1, is simple and yet capable of achieving satisfactory performance under the right conditions. Its major limitations are a relatively slow rate of convergence and a sensitivity to variations in the condition number of the correlation matrix of the tap inputs; the condition number of a Hermitian matrix is defined as the ratio of its largest eigenvalue to its smallest eigenvalue. Nevertheless, the LMS algorithm is highly popular and widely used in a variety of applications.

In a nonstationary environment, the orientation of the error-performance surface varies continuously with time. In this case, the LMS algorithm has the added task of continually tracking the bottom of the error-performance surface. Indeed, tracking will occur provided that the input data vary slowly compared to the learning rate of the LMS algorithm.

The stochastic gradient approach may also be pursued in the context of a lattice structure. The resulting adaptive filtering algorithm is called the gradient adaptive lattice (GAL) algorithm. In their own individual ways, the LMS and GAL algorithms are just two members of the stochastic gradient family of linear adaptive filters, although it must be said that the LMS algorithm is by far the most popular member of this family.

Least-Squares Estimation

The second approach to the development of linear adaptive filtering algorithms is based on the method of least squares. According to this method we minimize a cost function or index of performance that is defined as the sum of weighted error squares, where the error or residual is itself defined as the difference between some desired response and the actual filter output. The method of least squares may be formulated with block estimation or recursive estimation in mind. In block estimation the input data stream is arranged in the form of blocks of equal length (duration), and the filtering of input data proceeds on a block-by-block basis. In recursive estimation, on the other hand, the estimates of interest (e.g., tap weights of a transversal filter) are updated on a sample-by-sample basis. Ordinarily, a recursive estimator requires less storage than a block estimator, which is the reason for its much wider use in practice.

P517567.PDF [Page: 13 of 78]

Table 1: Summary of the LMS Algorithm

Notations:

$\mathbf{u}(n)$ = tap-input vector at time n

$= [u(n), u(n-1), \cdots, u(n-M+1)]^T$

M = number of tap inputs

$d(n)$ = desired response at time n

$\hat{\mathbf{w}}(n) = [w_0(n), w_1(n), \ldots, w_{M-1}(n)]^T$

= tap-weight vector at time n

$y(n)$ = actual response of the tapped-delay-line filter

$= \hat{\mathbf{w}}^H(n)\mathbf{u}(n)$, where superscript H denotes Hermitian transposition

$e(n)$ = error signal

$= d(n) - y(n)$

Parameters: M = number of taps

μ = step-size parameter

$$0 < \mu < \frac{2}{\text{tap-input power}}$$

$$\text{tap-input power} = \sum_{k=0}^{M-1} E[|u(n-k)|^2]$$

Initialization: If prior knowledge on the tap-weight vector $\hat{\mathbf{w}}(n)$ is available, use it to select an appropriate value for $\hat{\mathbf{w}}(0)$. Otherwise, set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Data:

- Given: $\mathbf{u}(n)$ = M-by-1 tap-input vector at time n

  $d(n)$ = desired response at time n
- To be computed:

  $\hat{\mathbf{w}}(n+1)$ = estimate of tap-weight vector at time $n+1$

Computation: For $n = 0, 1, 2, \cdots$, compute

$$e(n) = d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n)$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n)$$

P517567.PDF [Page: 14 of 78]

Recursive least-squares (RLS) estimation may be viewed as a special case of Kalman filtering. A distinguishing feature of the Kalman filter is the notion of state, which provides a measure of all the inputs applied to the filter up to a specific instant of time. Thus, at the heart of the Kalman filtering algorithm we have a recursion that may be described in words as follows:

$$\begin{pmatrix}\text{updated value}\\ \text{of the}\\ \text{state}\end{pmatrix} = \begin{pmatrix}\text{old value}\\ \text{of the}\\ \text{state}\end{pmatrix} + \begin{pmatrix}\text{Kalman}\\ \text{gain}\end{pmatrix}\begin{pmatrix}\text{innovation}\\ \text{vector}\end{pmatrix}$$

where the innovation vector represents new information put into the filtering process at the time of the computation. For the present, it suffices to say that there is indeed a one-to-one correspondence between the Kalman variables and RLS variables. This correspondence means that we can tap the vast literature on Kalman filters for the design of linear adaptive filters based on recursive least-squares estimation.

We may classify the recursive least-squares family of linear adaptive filtering algorithms into three distinct categories, depending on the approach taken:

1. Standard RLS algorithm, which assumes the use of a transversal filter as the structural basis of the linear adaptive filter. Table 2 summarizes the standard RLS algorithm. Derivation of this algorithm relies on a basic result in linear algebra known as the matrix inversion lemma. Most importantly, it enjoys the same virtues and suffers from the same limitations as the standard Kalman filtering algorithm. The limitations include lack of numerical robustness and excessive computational complexity. Indeed, it is these two limitations that have prompted the development of the other two categories of RLS algorithms, described next.
2. Square-root RLS algorithms, which are based on QR-decomposition of the incoming data matrix. Two well-known techniques for performing this decomposition are the Householder transformation and the Givens rotation, both of which are data-adaptive transformations. At this point in the discussion, we need to merely say that RLS algorithms based on the Householder transformation or Givens rotation are numerically stable and robust. The resulting linear adaptive filters are referred to as square-root adaptive filters, because in a matrix sense they represent the square-root forms of the standard RLS algorithm
3. Fast RLS algorithms. The standard RLS algorithm and square-root RLS algorithms have a computational complexity that increases as the square of M, where M is the number of adjustable weights (i.e., the number of degrees of freedom) in the algorithm. Such algorithms are often referred to as $O(M^2)$ algorithms, where $O(\cdot)$ denotes "order of". By contrast, the LMS algorithm is an $O(M)$ algorithm, in that its computational complexity increases linearly with M. When M is large, the computational complexity of $O(M^2)$ algorithms may become objectionable from a hardware implementation point of view. There is therefore a strong motivation to modify the formulation of the RLS algorithm in such a way that the computational complexity assumes an $O(M)$ form. This objective is indeed achievable, in the case of temporal processing, first by virtue of the inherent redundancy in the Toeplitz structure of the input data matrix and, second, by exploiting this redundancy through the use of linear least-squares prediction in both the forward and backward directions. The resulting algorithms

P517567 PDF [Page: 15 of 78]

are known collectively as fast RLS algorithms; they combine the desirable characteristics of recursive linear least-squares estimation with an $O(M)$ computational complexity. Two types of fast RLS algorithms may be identified, depending on the filtering structure employed:

- Order-recursive adaptive filters, which are based on a lattice like structure for making linear forward and backward predictions.
- Fast transversal filters, in which the linear forward and backward predictions are performed using separate transversal filters.

Table 2: Summary of the RLS Algorithm

Notations:

$\mathbf{u}(n)$ = tap-input vector at time n

$= [u(n), u(n-1), \cdots, u(n-M+1)]^T$

M = number of tap inputs

$d(n)$ = desired response at time n

$\hat{\mathbf{w}}(n) = [\hat{w}_0(n), \hat{w}_1(n), \ldots, \hat{w}_{M-1}(n)]^T$

= tap-weight vector at time n

$\xi(n)$ = innovation (i.e., a priori error signal) at time n

λ = exponential weighting factor

$\mathbf{k}(n)$ = gain vector at time n

$\mathbf{P}(n)$ = weight-error correlation matrix

Initialize the algorithm by setting

$\mathbf{P}(0) = \delta^{-1}\mathbf{I}$, δ = small positive constant

$\hat{\mathbf{w}}(0) = \mathbf{0}$

For each instant of time, $n = 1, 2, \cdots$, compute

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)}$$

$$\xi(n) = d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n)$$

$$\hat{\mathbf{w}}(n) = \hat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)$$

$$\lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}(n)\mathbf{P}(n-1)$$

Certain (but not all) realizations of order-recursive adaptive filters are known to be numerically stable, whereas fast transversal filters suffer from a numerical stability problem and therefore require some form of stabilization for them to be of practical use.

P517567.PDF [Page: 16 of 78]

An introductory discussion of linear adaptive filters would be incomplete without saying something about their tracking behavior. In this context, we note that stochastic gradient algorithms such as the LMS algorithm are model-independent; generally speaking, we would expect them to exhibit good tracking behavior, which indeed they do. In contrast, RLS algorithms are model-dependent; this, in turn, means that their tracking behavior may be inferior to that of a member of the stochastic gradient family, unless care is taken to minimize the mismatch between the mathematical model on which they are based and the underlying physical process responsible for generating the input data.

How to Choose an Adaptive Filter

Given the wise variety of adaptive filters available to a system designer, how can a choice be made for an application of interest? Clearly, whatever, the choice, it has to be cost-effective. With this goal in mind, we may identify three important issues that require attention: computational cost, performance, and robustness. The use of computer simulation provides a good first step in undertaking a detailed investigation of these issues. We may begin by using the LMS algorithm as an adaptive filtering tool for the study. The LMS algorithm is relatively simple to implement. Yet it is powerful enough to evaluate the practical benefits that may result from the application of adaptivity to the problem at hand. Moreover, it provides a practical frame of reference for assessing any further improvement that may be attained through the use of more sophisticated adaptive filtering algorithms. Finally, the study must include tests with real-life data, for which there is no substitute.

Practical applications of adaptive filtering are very diverse, with each application having peculiarities of its own. The solution for one application may not be suitable for another. Nevertheless, to be successful we have to develop a physical understanding of the environment in which the filter has to operate and thereby relate to the realities of the application of interest.

5. Real and Complex Forms of Adaptive Filters

In the development of adaptive filtering algorithms, regardless of their origin, it is customary to assume that the input data are in baseband form. The term "baseband" is used to designate the band of frequencies representing the original (message) signal as generated by the source of information.

In such applications as communications, radar, and sonar, the information-bearing signal component of the receiver input typically consists of a message signal modulated onto a carrier wave. The bandwidth of the message signal is usually small compared to the carrier frequency, which means that the modulated signal is a narrow-band signal. To obtain the baseband representation of a narrow-band signal, the signal is translated down in frequency in such a way that the effect of the carrier wave is completely removed, yet the information content of the message signal is fully preserved. In general, the baseband signal so obtained is complex. In other words, a sample $u(n)$ of the signal may be written as

$$u(n) = u_I(n) + ju_Q(n) \tag{6}$$

P517567 PDF [Page: 17 of 78]

where $u_I(n)$ is the in-phase (real) component, and $u_Q(n)$ is the quadrature (imaginary) component. Equivalently, we may express $u(n)$ as

$$u(n) = |u(n)|e^{j\phi(n)} \tag{7}$$

where $|u(n))|$ is the magnitude and $\phi(n)$ is the phase angle.

The LMS and RLS algorithms summarized in Tables 1 and 2 assume the use of complex signals. Adaptive filtering algorithm so described are said to be in complex form. The important virtue of complex adaptive filters is that they preserve the mathematical formulation and elegant structure of complex signals encountered in the aforementioned areas of application.

If the signals to be processed are real, we naturally use the real form of the adaptive filtering algorithm of interest. Given the complex form of an adaptive filtering algorithm, it is straightforward to deduce the corresponding real form of the algorithm. Specifically, we do two things:

1. The operation of complex conjugation, wherever in the algorithm, is simply removed.
2. The operation of Hermitian transposition (i.e., conjugate transposition) of a matrix, wherever in the algorithm, is replaced by ordinary transposition.

Simply put, complex adaptive filters include real adaptive filters as special cases.

6. Nonlinear Adaptive Systems: Neural Networks

The theory of linear optimum filters is based on the mean-square error criterion. The Wiener filter that results from the minimization of such a criterion, and which represents the goal of linear adaptive filtering for a stationary environment, can only relate to second-order statistics of the input data and no higher. This constraint limits the ability of a linear adaptive filter to extract information from input data that are non-Gaussian. Despite its theoretical importance, the existence of Gaussian noise is open to question (Johnson and Rao, 1990). Moreover, non-Gaussian processes are quite common in many signal processing applications encountered in practice. The use of a Wiener filter or a linear adaptive filter to extract signals of interest in the presence of such non-Gaussian processes will therefore yield suboptimal solutions. We may overcome this limitation by incorporating some form of nonlinearity in the structure of the adaptive filter to take care of higher-order statistics. Although by so doing, we no longer have the Wiener filter as a frame of reference and so complicate the mathematical analysis, we would expect to benefit in two significant ways: improving learning efficiency and a broadening of application areas.

In this section, we describe an important class of nonlinear adaptive system commonly known as artificial neural networks or just simply neural networks. This terminology is derived from analogy with biological neural networks that make up the human brain.

P517567.PDF [Page: 18 of 78]

A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

(i) Knowledge is acquired by the network through a learning process.
(ii) Interconnection strengths known as synaptic weights are used to store the knowledge.

Basically, learning is a process by which the free parameters (i.e., synaptic weights and bias levels) of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place. Specifically, learning machines may be classified as follows:

- Learning with a teacher, also referred to as supervised learning.
- Learning without a teacher

This second class of learning machines may also be subdivided into:

- Reinforcement learning
- Unsupervised learning or self-organized learning

In the subsequent sections of this chapter, we will describe the important aspects of these learning machines and highlight the algorithms involved in their designs. For a detailed treatment of the subject, see Haykin (1999); this book has an up-to-date bibliography that occupies 41 pages of references.

In the context of adaptive signal processing applications, neural networks offer the following advantages:

- Nonlinearity, which makes it possible to account for the nonlinear behavior of physical phenomena responsible for generating the input data
- The ability to approximate any prescribed input-output mapping of a continuous nature
- Weak statistical assumptions about the environment, in which the network is embedded
- Learning capability, which is accomplished by undertaking a training session with input-output examples that are representative of the environment
- Generalization, which refers to the ability of the neural network to provide a satisfactory performance in response to test data never seen by the network before
- Fault tolerance, which means that the network continues to provide an acceptable performance despite the failure of some neurons in the network
- VLSI implementability, which exploits the massive parallelism built into the design of a neural network.

This is indeed an impressive list of attributes, which accounts for the widespread interest in the use of neural networks to solve signal-processing tasks that are too difficult for conventional (linear) adaptive filters.

P517567.PDF [Page: 19 of 78]

Supervised Learning

This form of learning assumes the availability of a labeled (i.e., ground-truthed) set of training data made up of N input-output examples:

$$T = \{(\mathbf{x}_i, d_i)\}_{i=1}^{N} \tag{8}$$

where $\mathbf{x}_i$ = input vector of ith example

d_i = desired (target) response of ith example, assumed to be scalar for convenience of presentation

N = Sample size

Given the training sample T, the requirement is to compute the free parameters of the neural network so that the actual output y_i of the neural network due to $\mathbf{x}_i$ is close enough to d_i for all i in a statistical sense. For example, we may use the mean-squared error

$$E(n) = \frac{1}{N} \sum_{i=1}^{N} (d_i - y_i)^2 \tag{9}$$

as the index of performance to be minimized.

Multilayer Perceptrons and Back-Propagation Learning

The back-propagation algorithm has emerged as the workhorse for the design of a special class of layered feedforward networks known as multilayer perceptrons. As shown in the block diagram of Fig. 6, a multilayer perceptron consists of the following:

- Input layer of nodes, which provide the means for connecting the neural network to the source(s) of signals driving the network.
- One or more hidden layers of processing units, which act as "feature detectors".
- Output layer of processing units, which provide one final stage of computation and thereby produce the response of the network to the signals applied to the input layer.

The processing units are commonly referred to as artificial neurons or just neurons. Typically, a neuron consists of a linear combiner with a set of adjustable synaptic weights, followed by a nonlinear activation function, as depicted in Fig. 7.

Two commonly used forms of the activation function $\varphi(\cdot)$ are shown in Fig. 8. The first one, shown in Fig. 8a, is called the hyperbolic function, which is defined by

$$\varphi(v) = \tanh(v)$$

P517567.PDF [Page: 20 of 78]

$$= \frac{1 - \exp(-2v)}{1 + \exp(-2v)} \tag{10}$$

The second one, shown in Fig. 8b, is called the logistic function, which is defined by

$$\varphi(v) - \frac{1}{1 + \exp(-v)} \tag{11}$$

From these definitions, we readily see that the logistic function is of a unipolar form that is nonsymmetric, whereas the hyperbolic function is bipolar that is antisymmetric.

The training of an MLP is usually accomplished by using the back-propagation (BP) algorithm that involves two phases (Werbos, 1974; Rumelhart et al., 1986):

- Forward phase. During this phase the free parameters of the network are fixed, and the input signal is propagated through the network of Fig. 6 layer by layer. The forward phase finishes with the computation of an error signal defined as the difference between a desired response and the actual output produced by the network in response to the signals applied to the input layer.
- Backward phase. During this second phase, the error signal e_i is propagated through the network of Fig. 6 in the backward direction, hence the name of the algorithm. It is during this phase that adjustments are applied to the free parameters of the network so as to minimize the error e_i in a statistical sense.

Back-propagation learning may be implemented in one of two basic ways, as summarized here:

(i) Sequential mode (also referred to as the pattern mode, on-line mode, or stochastic mode):
In this mode of BP learning, adjustments are made to the free parameters of the network on an example-by-example basis. The sequential mode is best suited for pattern classification.

(ii) Batch mode:
In this second mode of BP learning, adjustments are made to the free parameters of the network on an epoch-by-epoch basis, where each epoch consists of the entire set of training examples.The batch mode is best suited for nonlinear regression.

The back-propagation learning algorithm is simple to implement, and computationally efficient in that its complexity is linear in the synaptic weights of the network. However, a major limitation of the algorithm is that it can be excruciatingly slow, particularly when we have to deal with a difficult learning task that requires the use of a large network.

Traditionally, the derivation of the back-propagation algorithm is done for real-valued data. This derivation may be extended to complex-valued data by permitting the free parameters of the multilayer perceptron assume complex values. However, in the latter case, care has to be exercised in how the activation function is handled for complex-valued inputs. For a detailed derivation of the complex back-propagation algorithm, see Haykin (1996).

P517567.PDF [Page: 21 of 78]

In any event, we may try to make back-propagation learning perform better by invoking the following list of neuristics:

- Use neurons with antisymmetric activation functions (e.g., hyperbolic tangent function) in preference to nonsymmetric activation functions (e.g., logistic function). Figure 8 shows examples of these two forms of activation functions.
- Shuffle the training examples after the presentation of each epoch; an epoch involves the presentation of the entire set of training examples to the network.
- Follow an easy-to-learn example with a difficult one.
- Preprocess the input data so as to remove the mean and decorrelate the data.
- Arrange for the neurons in the different layers to learn at essentially the same rate. This may be attained by assigning a learning-rate parameter to neurons in the last layers that is smaller than those at the front end.
- Incorporate prior information into the network design whenever it is available.

One other heuristic that deserves to be mentioned relates to the size of the training set, N, for a pattern classification task. Given a multilayer perceptron with a total number of synaptic weights including bias levels, denoted by W, a rule of thumb for selecting N is

$$N = O\left(\frac{W}{\varepsilon}\right) \tag{12}$$

where O denotes "the order of", and ε denotes the fraction of classification errors permitted on test data. For example, with an error of 10% the number of training examples needed should be about 10 times the number of synaptic weights in the network.

Supposing that we have chosen a multilayer perceptron to be trained with the back-propagation algorithm, how do we determine when it is "best" to stop the training session? How do we select the size of individual hidden layers of the MLP? The answers to these important questions may be gotten though the use of a statistical technique known as cross-validation, which proceeds as follows:

- The set of training examples is split into two parts:
  - (i) Estimation subset used for training of the model
  - (ii) Validation subset used for evaluating the model performance
- The network is finally tuned by using the entire set of training examples and then tested on test data not seen before.

Radial-basis function (RBF) networks

Another popular layered feedforward network is the radial-basis function (RBF) network, whose structure is shown in Fig. 9. RBF networks use memory-based learning for their design. Specifically, learning is viewed as a curve-fitting problem in high-dimensional space (Broomhead and Lowe, 1988; Poggio and Girosi, 1990):

P517567.PDF [Page: 22 of 78]

(i) Learning is equivalent to finding a surface in a multidimensional space that provides a best fit to the training data.
(ii) Generalization (i.e., response of the network to input data not seen before) is equivalent to the use of this multidimensional surface to interpolate the test data.

A commonly used formulation of the radial basis functions, which constitute the hidden layer, is based on the Gaussian function. To be specific, let $\mathbf{u}$ denote the signal vector applied to the input layer and $\mathbf{u}_i$ denote the center of the Gaussian function assigned to hidden unit i. We may then define the corresponding radial basis function as

$$\varphi(\|\mathbf{u}-\mathbf{u}_i\|) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{u}_i\|^2}{2\sigma^2}\right), \qquad i = 1, 2, \cdots, K \tag{13}$$

where the symbol $\|\mathbf{u}-\mathbf{u}_i\|$ denotes the Euclidean distance between the vectors $\mathbf{u}$ and $\mathbf{u}_i$, and σ^2 is the width common to all K radial basis functions. (Each radial basis function may also be permitted to have a different width, but such a generalization results in increased complexity.) On this basis, we may define the input-output mapping realized by the RBF network (assuming a single output)

$$y = \sum_{i=1}^{K} w_i \varphi(\|\mathbf{u}-\mathbf{u}_i\|)$$

$$= \sum_{i=1}^{K} w_i \exp\left(-\frac{\|\mathbf{u}-\mathbf{u}_i\|^2}{2\sigma^2}\right) \tag{14}$$

where the set of weights $\{w_i\}_{i=1}^{K}$ constitutes the output layer. Equation (14) represents a linear mixture of Gaussian functions.
RBF networks differ from multilayer perceptrons in some fundamental respects:

- RBF networks are local approximators whereas multilayer perceptrons are global approximators.
- RBF networks have a single hidden layer whereas multilayer perceptrons can have any number of hidden layers.
- The output layer of a RBF network is always linear whereas in a multilayer perceptron it can be linear or nonlinear.
- The activation function of the hidden layer in an RBF network computes the Euclidean distance between the input signal vector and a parameter vector of the network, whereas the activation function of a multilayer perceptron computes the inner product between the input signal vector and the pertinent synaptic weight vector.

P517567.PDF [Page: 23 of 78]

The use of a linear output layer in an RBF network may be justified in light of Cover's theorem on the separability of patterns. According to this theorem, provided that the transformation from the input space to the feature (hidden) space is nonlinear and the dimensionality of the feature space is high compared to that of the input (data) space, then there is a high likelihood that a nonseparable pattern classification task in the input space is transformed into a linearly separable one in the feature space.

Design methods for RBF networks include the following:

(i) Random selection of fixed centres (Broomhbead and Lowe, 1988)
(ii) Self-organized selection of centres (Moody and Darken, 1989)
(iii) Supervised selection of centres (Poggio and Girosi, 1990)
(iv) Regularized interpolation exploiting the connection between an RBF network and the Watson-Nadaraya regression kernel (Yee, 1998).

Support Vector Machines

Support vector machine (SVM) theory provides the most principled approach to the design of neural networks, eliminating the need for domain knowledge (Vapnik, 1998). SVM theory applies to pattern classification, regression, or density estimation using an RBF network (depicted in Fig. 9) or an MLP with a single hidden layer (depicted in Fig. 6).

Unlike back-propagation learning, different cost functions are used for pattern classification and regression. Most importantly, the use of SVM learning eliminates the need for how to select the size of the hidden layer in an MLP or RBF network. In the latter case, it also eliminates the need for how to specify the centers of the RBF units in the hidden layer.

Simply stated, support vectors are those data points (for the linearly separable case) that are the most difficult to classify and optimally separated from each other.

In a support vector machine, the selection of basis functions is required to satisfy Mercer's theorem; that is, each basis function is in the form of a positive definite inner-product kernel (assuming real-valued data):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \underset{\sim}{\varphi}^T(\mathbf{x}_i)\underset{\sim}{\varphi}(\mathbf{x}_j) \tag{15}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are input vectors for examples i and j, and $\underset{\sim}{\varphi}(\mathbf{x}_i)$ is the vector of hidden-unit outputs for inputs $\mathbf{x}_i$. The hidden (feature) space is chosen to be of high dimensionality so as to transform a nonlinear separable pattern classification problem into a linearly separable one. Most importantly, however, in a pattern-classification task, for example, the support vectors are selected by the SVM learning algorithm so as to maximize the margin of separation between classes.

The curse-of-dimensionality problem, which can plague the design of multilayer perceptrons and RBF networks, is avoided in support vector machines through the use of quadratic programming.

P517567.PDF [Page: 24 of 78]

This technique, based directly on the input data, is used to solve for the linear weights of the output layer (Vapnik, 1998).

Unsupervised Learning

Turning next to unsupervised learning, adjustment of synaptic weights may be carried through the use of neurobiological principles such as Hebbian learning and competitive learning, or information-theoretic principles. In this section we will describe specific applications of these three approaches.

Principal components analysis

According to Hebb's postulate of learning, the change in synaptic weight Δw_{ji} of a neural network is defined by (for real-valued data)

$$\Delta w_{ji} = \eta x_i y_j \tag{16}$$

where η = learning-rate parameter
x_i = input (presynaptic) signal
y_j = output (postsynaptic) signal

Principal component analysis (PCA) networks use a modified form of this self-organized learning rule. To begin with, consider a linear neuron designed to operate as a maximum eigenfilter; such a neuron it referred to as Oja's neuron (Oja, 1982). It is characterized as follows:

$$\Delta w_{ji} = \eta\ y_j(x_i - y_j w_{ji}) \tag{17}$$

where the term $-\eta\ y_j^2 w_{ji}$ is added to stabilize the learning process. As the number of iterations approaches infinity, we find the following:

(i) The synaptic weight vector of neuron j approaches the eigenvector associated with the largest eigenvalue λ_{max} of the correlation matrix of the input vector (assumed to be of zero mean).
(ii) The variance of the output of neuron j approaches the largest eigenvalue λ_{max}.

The generalized Hebbian algorithm (GHA), due to Sanger (1989), is a straightforward generalization of Oja's neuron for the extraction of any desired number of principal components.

Self-organizing Maps

In a self-organizing map (SOM), due to Kohonen (1997), the neurons are placed at the nodes of a lattice, and they become selectively tuned to various input patterns (vectors) in the course of a competitive learning process. The process is characterized by the formation of a topographic map in which the spatial locations (i.e., coordinates) of the neurons in the lattice correspond to intrinsic

P517567.PDF [Page: 25 of 78]

features of the input patterns. Figure 10 illustrates the basic idea of a self-organizing map, assuming the use of a two-dimensional lattice of neurons as the network structure.

In reality, the SOM belongs to the class of vector coding algorithms (Luttrell, 1989). That is, a fixed number of codewords are placed into a higher-dimensional input space, thereby facilitating data compression.

An integral feature of the SOM algorithm is the neighborhood function centred around a neuron that wins the competitive process. The neighborhood function starts by enclosing the entire lattice initially, and is then allowed to shrink gradually until it encompasses the winning neuron.

The algorithm exhibits two distinct phases in its operation:

(i) Ordering phase, during which the topological ordering of the weight vectors takes place.
(ii) Convergence phase, during which the computational map is fine tuned.

The SOM Algorithm exhibits the following properties:

1. Approximation of the continuous input space by the weight vectors of the discrete lattice.
2. Topological ordering exemplified by the fact that the spatial location of a neuron in the lattice corresponds to a particular feature of the input pattern.
3. The feature map computed by the algorithm reflects variations in the statistics of the input distribution.
4. SOM may be viewed as a nonlinear form of principal components analysis.

Information-theoretic Models

Mutual information, defined in accordance with Shannon's information theory, provides the basis of a powerful approach for self-organized learning. The theory is embodied in the maximum mutual information (Infomax) principle, due to Linsker (1988), which may be stated as follows:

> The transformation of a random vector **X** observed in the input layer of a neural network to a random vector **Y** produced in the output layer should be so chosen that the activities of the neurons in the output layer jointly maximize information about the activities in the input layer. The objective function to be maximized is the mutual information $I(\mathbf{Y};\mathbf{X})$ between **X** and **Y**.

The Infomax principle finds applications in the following areas:

- Design of self-organized models and feature maps (Linsker, 1989).
- Discovery of properties of a noisy sensory input exhibiting coherence across both space and time (first variant of Infomax due to Becker and Hinton, 1992).
- Dual image processing designed to maximize the spatial differentiation between the corresponding regions of two separate images (views) of an environment of interest as in radar polarimetry (second variant of Infomax due to Ukrainec and Haykin, 1996).

P517567.PDF [Page: 26 of 78]

- Independent components analysis (ICA) for blind source separation (due to Barlow, 1989); see also Comon (1994). ICA may be viewed as the third variant of Infomax (Haykin, 1999).

Temporal Processing Using Feedforward Networks

Time is an essential dimension of learning. We may incorporate time into the design of a neural network implicitly or explicitly. A straightforward method of implicit representation of time is to add a short-term memory structure at the input end of a static neural network (e.g., multilayer perceptron), as illustrated in Fig. 11. This configuration is called a focused time-lagged feedforward network (TLFN). Focused TLFNs are limited to stationary dynamical processes.

To deal with nonstationary dynamical processes, we may use distributed TLFNs where the effect of time is distributed at the synaptic level throughout the network. One way in which this may be accomplished is to use finite-duration impulse response (FIR) filters to implement the synaptic connections of an MLP. The training of a distributed TLFN is naturally a more difficult proposition than the training of a focused TLFN. Whereas we may use the ordinary back-propagation algorithm to train a focused TLFN, we have to extend the back-propagation algorithm to cope with the replacement of a synaptic weight in the ordinary MLP by a synaptic weight vector. This extension is referred to as the temporal back propagation algorithm due to Wan (1994).

Dynamically Driven Recurrent Networks

Another practical way of accounting for time in a neural network is to employ feedback at the local or global level. Neural networks so configured are referred to as recurrent networks.

We may identify two classes of recurrent networks:

1. Autonomous recurrent networks exemplified by the Hopfield network (Hopfield, 1982) and brain-state-in-a-box (BSB) model. These networks are well suited for building associative memories, each with its own domain of applications. Figure 12 shows an example of a Hopfield network involving the use of four neurons.
2. Dynamically driven recurrent networks, which are well suited for input-output mapping functions that are temporal in character.

A powerful approach for the design of dynamically driven recurrent networks with the goal of solving an input-output mapping task is to build on the state-space approach of modern control theory (Sontag, 1990). Such an approach is well suited for the two network configurations shown in Figs. 13 and 14, which are respectively referred to as a nonlinear autoregressive with exogeneous inputs (NARX) model and recurrent multilayer perceptron (RMLP).

To design a dynamically driven recurrent network for input-output mapping, we may use any one of the following approaches:

P517567.PDF [Page: 27 of 78]

- Back-propagation through time (BPTT), which involves unfolding the temporal operation of the recurrent network into a layered feedforward network (Werbos, 1990). This unfolding facilitates the application of the ordinary back-propagation algorithm.
- Real-time recurrent learning, in which adjustments are made (using a gradient-descent method) to the synaptic weights of a fully connected recurrent network in real time (Williams and Zipser, 1989).
- Extended Kalman filter (EKF), which builds on the classic Kalman filter theory to compute the synaptic weights of the recurrent network. Two versions of the algorithm are available (Feldkamp and Puskorius, 1998):
    - Decoupled EKF
    - Global EKF

The decoupled EKF algorithm is computationally less demanding but somewhat less accurate than the global EKF algorithm.

A serious problem that can arise in the design of a dynamically driven recurrent network is the vanishing gradients problem. This problem pertains to the training of a recurrent network to produce a desired response at the current time that depends on input data in the distant past (Bengio et al., 1994). It makes the learning of long-term dependencies in gradient-based training algorithms difficult if not impossible in certain cases. To overcome the problem, we may use the following methods:

(i) Extended Kalman filter (encompassing second-order information) for training.
(ii) Elaborate optimization methods such as pseudo-Newton and simulated annealing (Bengio et al., 1994).
(iii) Use of long time delays in the network architecture (Giles et al., 1997).
(iv) Hierarchially structuring of the network in multiple levels associated with different time scales (El Hihi and Bengio, 1996).
(v) Use of gating units to circumvent some of the nonlinearities (Hochreiter and Schmidhuber, 1997).

7. Applications

The ability of an adaptive filter to operate satisfactorily in an unknown environment and track time variations of input statistics make the adaptive filter a powerful device for signal-processing and control applications. Indeed, adaptive filters have been successfully applied in such diverse fields as communications, radar, sonar, seismology, and biomedical engineering. Although these applications are indeed quite different in nature, nevertheless, they have one basic common feature: an input vector and a desired response are used to compute an estimation error, which is in turn used to control the values of a set of adjustable filter coefficients. The adjustable coefficients may take the form of tap weights, reflection coefficients, rotation parameters, or synaptic weights, depending on the filter structure employed. However, the essential difference between the various applications of adaptive filtering arises in the manner in which the desired response is extracted. In this context, we may distinguish four basic classes of adaptive filtering applications, as depicted in Fig. 15. For convenience of presentation, the following notations are used in this figure

P517567.PDF [Page: 28 of 78]

u = input applied to the adaptive filter
y = output of the adaptive filter
d = desired response
$e = d - y$ = estimation error.

The functions of the four basic classes of adaptive filtering applications depicted herein are as follows:

I. Identification [Fig. 15(a)]. The notion of a mathematical model is fundamental to sciences and engineering. In the class of applications dealing with identification, an adaptive filter is used to provide a linear model that represents the best fit (in some sense) to an unknown plant. The plant and the adaptive filter are driven by the same input. The plant output supplies the desired response for the adaptive filter. If the plant is dynamic in nature, the model will be time varying.

II. Inverse modeling [Fig. 15(b)]. In this second class of applications, the function of the adaptive filter is to provide an inverse model that represents the best fit (in some sense) to an unknown noisy plant. Ideally, in the case of a linear system, the inverse model has a transfer function equal to the reciprocal (inverse) of the plant's transfer function, such that the combination of the two constitutes an ideal transmission medium. A delayed version of the plant (system) input constitutes the desired response for the adaptive filter. In some applications, the plant input is used without delay as the desired response.

III. Prediction [Fig. 15(c)]. Here the function of the adaptive filter is to provide the best prediction (in some sense) of the present value of a random signal. The present value of the signal thus serves the purpose of a desired response for the adaptive filter. Past values of the signal supply the input applied to the adaptive filter. Depending on the application of interest, the adaptive filter output or the estimation (prediction) error may serve as the system output. In the first case, the system operates as a predictor; in the latter case, it operates as a prediction-error filter.

IV. Interference canceling [Fig. 15(d)]. In this final class of applications, the adaptive filter is used to cancel unknown interference contained (alongside an information-bearing signal component) in a primary signal, with the cancelation being optimized in some sense. The primary signal serves as the desired response for the adaptive filter. A reference (auxiliary) signal is employed as the input toe the adaptive filter. The reference signal is derived from a sensor or set of sensors located in relation to the sensor(s) supplying the primary signal in such a way that the information-bearing signal component is weak or essentially undesirable.

In Table 3 we have listed some applications that are illustrative of the four basic classes of adaptive filtering applications. These applications, totaling twelve, are drawn from the fields of control systems, seismology, electrocardiography, communications, and radar. A selected number of these applications are described individually in the remainder of this section.

P517567.PDF [Page: 29 of 78]

System Identification

System identification is the experimental approach to the modeling of a process or a plant (Goodwin and Payne, 1977; Ljung and Söderström, 1983; Åström and Eykhoff, 1971). It involves the following steps: experimental planning, the selection of a model structure, parameter estimation, and model validation. The procedure of system identification, as pursued in practice, is iterative in nature in that we may have to go back and forth between these steps until a satisfactory model is built. Here we discuss briefly the idea of adaptive filtering algorithms for estimating the parameters of an unknown plant modeled as a transversal filter.

Suppose we have an unknown dynamic plant that is linear and time varying. The plant is characterized by a *real-valued* set of discrete-time measurements that describe the variation of the plant output in response to a known stationary input. The requirement is to develop an on-line transferal filter model for this plant, as illustrated in Fig. 16. The model consists of a finite number of unit-delay elements and a corresponding set of adjustable parameters (tap weights).
Let the available input signal at time n be denoted by the set of samples:

Table 3: Applications of Adaptive Filters

Class of adaptive filtering	Application
I. Identification	System identification Layered earth modeling
II. Inverse modeling	Predictive deconvolution Adaptive equalization Blind equalization
III. Prediction	Linear predictive coding Adaptive differential pulse-code modulation Autoregressive spectrum analysis Signal detection
IV. Interference canceling	Adaptive noise canceling Echo cancelation Adaptive beamforming

$u(n), u(n-1), \cdots, u(n-M+1)$, where M is the number of adjustable parameters in the model. This input signal is applied simultaneously to the plant and the model. Let their respective outputs be denoted by $d(n)$ and $y(n)$. The plant output $d(n)$ serves the purpose of a desired response for the adaptive filtering algorithm employed to adjust the model parameters. The model output is given by

$$y(n) = \sum_{k=0}^{M-1} \hat{w}_k(n) u(n-k) \tag{18}$$

P517567.PDF [Page: 30 of 78]

where $\hat{w}_0(n)$, $\hat{w}_1(n)$, $\cdots$, and $\hat{w}_{M-1}(n)$ are the estimated model parameters. The model output $y(n)$ is compared with the plant output $d(n)$. The difference between them, $d(n) - y(n)$, defines the modeling (estimation) error. Let this error be denoted by $e(n)$.

Typically, at time n, the modeling error $e(n)$ is nonzero, implying that the model deviates from the plant. In an attempt to account for this deviation, the error $e(n)$ is applied to an adaptive control algorithm. The samples of the input signal, $u(n)$, $u(n-1)$, $\cdots$, $u(n-M+1)$, are also applied to the algorithm. The combination of the transversal filter and the adaptive control algorithm constitutes the adaptive filtering algorithm. The algorithm is designed to control the adjustments made in the values of the model parameters. As a result, the model parameters assume a new net of values for use on the next iteration. Thus, at time $n + 1$, a new model output is computed, and with it a new value for the modeling error. The operation described is then repeated. This process is continued for a sufficiently large number of iterations (starting from time $n = 0$), until the deviation of the model from the plant, measured by the magnitude of the modeling error $e(n)$, becomes sufficiently small in some statistical sense.

When the plant is time varying, the plant output is nonstationary, and so is the desired response presented to the adaptive filtering algorithm. In such a situation, the adaptive filtering algorithm has the task of not only keeping the modeling error small but also continually tracking the time variations in the dynamics of the plant.

Spectrum Estimation

The *power spectrum* provides a quantitative measure of the second-order statistics of a discrete-time stochastic process as a function of frequency. In parametric spectrum analysis, we evaluate the power spectrum of the process by assuming a model for the process. In particular, the process is modeled as the output of a linear filter that is excited by a white-noise process, as in Fig. 16. By definition, a white-noise process has a constant power spectrum. A model that is of practical utility is the autoregressive (AR) model, in which the transfer function of the filter is assumed to consist of poles only. Let this transfer function be denoted by

$$H(e^{j\omega}) = \frac{1}{1 + a_1 e^{j\omega} + \ldots + a_M e^{-jM\omega}}$$

$$= \frac{1}{1 + \sum_{k=1}^{M} a_k e^{-jk\omega}} \tag{19}$$

where the a_k are called the autoregressive (AR) parameters, and M is the model order. Let σ_v^2 denote the constant power spectrum of the white-noise process $v(n)$ applied to the filter input. Accordingly, the power spectrum of the filter output $u(n)$ equals

$$S_{AR}(\omega) = \sigma_v^2 |H(e^{j\omega})|^2 \tag{20}$$

P517567.PDF [Page: 31 of 78]

We refer to $S_{AR}(\omega)$ as the autoregressive (AR) power spectrum. Equation (19) assumes that the AR process $u(n)$ is real, in which case the AR parameters themselves assume real values.

When the AR model is time varying, the model parameters become time dependent, as shown by $a_1(n), a_2(n), \ldots, a_M(n)$. In this case, we express the power spectrum of the time-varying AR process as

$$S_{AR}(\omega, n) = \frac{\sigma_v^2}{\left|1 + \sum_{k=1}^{M} a_k e^{-jk\omega}\right|} \tag{21}$$

We may determine the AR parameters of the time-varying model by applying $u(n)$ to an adaptive prediction-error filter, as indicated in Fig. 18. The filter consists of a transversal filter with adjustable tap weights. In the adaptive scheme of Fig. 18, the prediction error produced at the output of the filter is used to control the adjustments applied to the tap weights of the filter. The adaptive AR model provides a practical means for measuring the instantaneous frequency of a frequency-modulated process. In particular, we may do this by measuring the frequency at which the AR power spectrum $S_{AR}(\omega,n)$ attains its peak value for varying time n.

Signal Detection

The detection problem, that is, the problem of detecting an information-bearing signal in noise, may be viewed as one of hypothesis testing with deep roots in statistical decision theory (Van Trees, 1968). In the statistical formulation of hypothesis testing, there are two criteria of most interest: the Bayes criterion and the Neyman-Pearson criterion. In the Bayes test, we minimize the average cost or risk of the experiment of interest, which incorporates two sets of parameters: (1) a priori probabilities that represent the observer's information about the source of information before the experiment is conducted, and (2) a set of costs assigned to the various possible courses of action. As such, the Bayes criterion is directly applicable to digital communications. In the Neyman-Pearson test, on the other hand, we maximize the probability of detection subject to the constraint that the probability of false alarm does not exceed some preassigned value. Accordingly, the Neyman-Pearson criterion is directly applicable to radar or sonar. An idea of fundamental importance that emerges in hypothesis testing is that, for a Bayes criterion or Neyman-Pearson criterion, the optimum test consists of two distinct operations: (1) processing the observed data to compute a test statistic called the likelihood ratio, and (2) computing the likelihood ratio with a threshold to make a decision in favor of one of the two hypotheses. The choice of one criterion or the other merely affects the value assigned to the threshold. Let H_1 denote the hypothesis that the observed data consist of noise alone, and H_2 denote the hypothesis that the data consist of signal plus noise. The likelihood ratio is defined as the ratio of two maximum likelihood functions, the numerator assuming that hypothesis H_2 is true and the denominator assuming that hypothesis H_1 is true. If the likelihood ratio exceeds the threshold, the decision is made in favor of hypothesis H_2; otherwise, the decision is made in favor of hypothesis H_1.

P517567.PDF [Page: 32 of 78]

In simple binary hypothesis testing, it is assumed that the signal is known, and the noise is both white and Gaussian. In this case, the likelihood ratio test yields a matched filter (matched in the sense that its impulse response equals the time-reversed version of the known signal). When the additive noise is a colored Gaussian noise of known mean and correlation matrix, the likelihood ratio test yields a filter that consists of two sections: a whitening filter that transforms the colored noise component at the input into a white Gaussian noise process, and a matched filter that is matched to the new version of the known signal as modified by the whitening filter.

However, in some important operational environments such as communications, radar, and active sonar, there may be inadequate information on the signal and noise statistics to design a fixed optimum detector. For example, in a sonar environment it may be difficult to develop a precise model for the received sonar signal, one that would account for the following factors completely:

- Loss in the signal strength of a target echo from an object of interest (e.g., enemy vessel), due to oceanic propagation effects and reflection loss at the target
- Statistical variations in the additive reverberation component, produced by reflections of the transmitted signal from scatterers such as the ocean surface, ocean floor, biologies, and in homogeneities within the ocean volume
- Potential sources of noise such as biological, shipping, oil drilling, seismic, and oceanographic phenomena.

In situations of this kind, the use of adaptivity offers a powerful approach to solve difficult signal detection problems. The particular application we have chosen for our present discussion is the detection of a small radar target in sea clutter (i.e., radar backscatter from the ocean surface). The radar target is a small piece of ice called a growler, the portion of which that is visible above the sea surface is about the size of a grand piano. Recognizing that 90% of the volume of ice lies inside the water, a growler can indeed pose a threat to navigation in ice-infested waters as on the East Coast of Canada.

The detection problem described herein is further compounded by the nonstationary nature of both sea clutter and the target echo from the growler. The strategy we have chosen to solve this difficult signal detection problem reformulates it into a pattern classification problem for which neural networks are well suited.

Figure 19 shows a block diagram of the detection strategy described in (Haykin and Thomson, 1998; Haykin and Bhattacharya, 1997). It consists of three functional units:

- Time-frequency analyzer, which converts the time-varying waveform of the input signal into a picture with two coordinates, namely, time and frequency.
- Feature extractor, the purpose of which is to compress the two-dimensional data produced by the time-frequency analyzer by extracting a set of features that retain the essential frequency content of the original signal.
- Pattern classifier, which is trained to categorize the set of features applied to its input into two classes:
  (i) There is no target present (i.e., the input signal consists of clutter only).
  (ii) There is a target present (i.e., the input signal consists of target echo plus clutter).

P517567.PDF [Page: 33 of 78]

A signal-processing tool that is well suited for the application described herein is the Wigner-Ville distribution (WVD) (Cohen, 1995). The time-frequency map produced by this method is highly dependent on the nature of the input signal. If the input signal consists of clutter only, the resulting WVD picture is determined entirely by the time-frequency characteristics of sea clutter. Figure 20a shows a typical WVD picture due to sea clutter acting alone. On the other hand, if the input signal consists of a target echo plus sea clutter, the resulting WVD picture consists of three components: one due to the target echo, another due to clutter in the background, and a third one (commonly referred to as a "cross-product term") due to the interaction between these two components. Ordinarily, the cross-product terms are viewed as highly undesirable as they tend to complicate the spectral interpretation of WVD pictures; indeed, much effort has been expended in the literature to reduce the effects of cross-product terms. However, in the application of WVD to target detection described herein, the cross-product terms perform a useful service by enhancing the detection power of the method. In particular, cross-product terms are there to be seen only when a target is present; they disappear when the input signal is target free. Figure 20b shows a typical WVD picture pertaining to the combined presence of sea clutter and radar echo from a small growler. The zebra-like pattern (consisting of an alternating set of dark and light stripes) is due to the cross-product terms. The point to note here is that the target echo in the original input signal is hardly visible; yet it shows up ever so clearly in the WVD picture of Fig. 20b.

For the feature extraction, we may use principal components analysis, which was briefly described in Section 6. Finally, for the pattern classification we may use a multilayer perceptron trained with the back-propagation algorithm. Design details of these two functional units and those of the WVD are presented elsewhere (Haykin and Thomson 1998; Haykin and Bhattacharya, 1997). For the present discussion, it suffices to compare the receiver operating characteristics of this new radar detection strategy against those of an ordinary constant false alarm rate (CFAR) receiver. Figure 21 presents the results of this comparison using real-life data, which were collected at a site on the East Coast of Canada by means of an instrument-quality radar known as the IPIX radar (designed and built at McMaster University, Hamilton, Ontario). From this figure, we see that for probability of false alarm $P_{FA} = 10^{-3}$, we have the following values for probability of detection:

$$P_D = \begin{cases} 0.91 & \text{for adaptive receiver based on the detection strategy of Fig. 19} \\ 0.71 & \text{for Doppler CFAR receiver} \end{cases}$$

Target Tracking

The objective of tracking is to estimate the state of a target of interest by processing measurements obtained from the target through the use of sensors and other means. The measurements are noise-corrupted observables, which are related to the current state of the target. Typically, the state consists of kinematic components such as the position, velocity, and acceleration of a moving target.

To state the tracking problem in mathematical terms, let the vector $\mathbf{x}(n)$ denote the state of a target, the vector $\mathbf{u}(n)$ denote the (known) input or control signal, and the vector $\mathbf{y}(n)$ denote the

P517567.PDF [Page: 34 of 78]

corresponding measurements obtained from the target. We may then express the state-space equations of the system in its most generic setting as follows:

$$\mathbf{x}(n+1) = \mathbf{f}(n, \mathbf{x}(n), \mathbf{u}(n), \mathbf{v}_1(n)) \tag{22}$$

$$\mathbf{y}(n) = \mathbf{h}(n, \mathbf{x}(n), \mathbf{v}_2(n)) \tag{23}$$

where $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ are vector-valued functions and $\mathbf{v}_1(n)$ and $\mathbf{v}_2(n)$ are noise vectors. The time argument n indicates that both $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ are time varying. Equations (22) and (23) are referred to as the process and measurement equations, respectively. The issue of interest is to estimate the state vector $\mathbf{x}(n)$, given the measurement vector $\mathbf{y}(n)$.

When the process and measurement equations are both linear, and the process noise vector $\mathbf{v}_1(n)$ and measurement noise vector $\mathbf{v}_2(n)$ are both modeled as zero-mean, white, Gaussian processes that are statistical independent, the Kalman filter provides the optimum estimate of the state $\mathbf{x}(n)$, given $\mathbf{y}(n)$ (Bar-Shalom and Fortmann 1988). Optimality here refers to minimization of the mean-square error between the actual motion of the target and the track (i.e., state trajectory) estimated from the measurements associated with the target.

Unfortunately, in many of the radar and sonar target-tracking problems encountered in practice, the process and measurement equations are nonlinear and the noise processes corrupting the state and measured data are non-Gaussian. The traditional approach for dealing with nonlinear dynamics is to use the extended Kalman filter (EKF), the derivation of which assumes knowledge of the nonlinear functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ and maintains the Gaussian assumptions about the process and noise vectors. The EKF closely resembles a Kalman filter except for the fact that each step of the standard Kalman filtering algorithm is replaced by its linearized equivalent (Bar-Shalom and Fortmann, 1988). However, the EKF approach to target tracking suffers from the following drawbacks when it is applied to an environment with nonlinear dynamics:

1. Linearization of the vector-valued functions $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ can produce system instability if the time-steps are not sufficiently short in duration.
2. Linearization of the underlying dynamics requires the determination of two Jacobians (i.e., matrices of partial derivatives):
   - Jacobian of the vector-valued function $\mathbf{f}(\cdot)$, evaluated at the latest filtered estimate of the state at time n.
   - Jacobian of the vector-valued function $\mathbf{h}(\cdot)$, evaluated at the one-step predicted estimate of the state at time $n+1$.

   The determination of these two Jacobians may lead to computational difficulties.
3. The use of a short-time step to avoid system instability, combined with the determination of these two Jacobians, may impose a high computational overload on the system.

To overcome these shortcomings of the EKF, we may use the unscented Kalman filter (UKF) (Julier and Uhlmann, 1997; Wan et al., 1999), which is a generalization of the standard linear

P517567.PDF [Page: 35 of 78]

Kalman filter to systems whose process and measurement models are nonlinear. The UKF is preferable to the EKF for solving nonlinear filtering problems for two reasons:

1. The unscented Kalman filter is accurate to third-order for Gaussian-distributed process and measurement errors. For non-Gaussian distributions, the UKF is accurate to at least the second-order. Accordingly, the UKF provides better performance than the traditional EKF.
2. Unlike the EKF, the UKF does not require the computation of Jacobians pertaining to process and measurement equations. It is therefore simpler than the EKF in computational terms.

These are compelling reasons to reconsider the design of tracking systems for radar and sonar systems using the unscented Kalman filter.

Adaptive Noise Canceling

As the name implies, adaptive noise canceling relies on the use of noise canceling by subtracting noise from a received signal, an operation controlled in an adaptive manner for the purpose of improved signal-to-noise ratio. Ordinarily, it is inadvisable to subtract noise from a received signal, because such an operation could produce disastrous results by causing an increase in the average power of the output noise. However, when proper provisions are made, and filtering and subtraction are controlled by an adaptive process, it is possible to achieve a superior system performance compared to direct filtering of the received signal (Widrow et al., 1975b; Widrow and Stearns, 1985).

Basically, an adaptive noise canceler is a dual-input, closed-loop adaptive feedback system as illustrated in Fig. 22. The two inputs of the system are derived from a pair of sensors: a primary sensor and a reference (auxiliary) sensor. Specifically, we have the following:

1. The primary sensor receives an information-bearing signal $s(n)$ corrupted by additive noise $v_0(n)$, as shown by

$$d(n) = s(n) + v_0(n) \tag{24}$$

   The signal $s(n)$ and the noise $v_0(n)$ are uncorrelated with each other; that is,

$$E[s(n)v_0(n-k)] = 0 \quad \text{for all } k \tag{25}$$

   where $s(n)$ and $v_0(n)$ are assumed to be real valued.

2. The reference sensor receives a noise $v_1(n)$ that is uncorrelated with the signal $s(n)$ but correlated with the noise $v_0(n)$ in the primary sensor output in an unknown way; that is

$$E[s(n)v_1(n-k)] = 0 \quad \text{for all } k \tag{26}$$

   and

P517567.PDF [Page: 36 of 78]

$$E[v_0(n)v_1(n-k)] = p(k) \tag{27}$$

where, as before, the signals are real valued and $p(k)$ is an unknown cross-correlation for lag k. The reference signal $v_1(n)$ is processed by an adaptive filter to produce the output signal:

$$y(n) = \sum_{k=0}^{M-1} \hat{w}_k(n)v_1(n-k) \tag{28}$$

where $\hat{w}_k(n)$ are the adjustable (real) tap weights of the adaptive filter. The filter output $y(n)$ is subtracted from the primary signal $d(n)$, serving as the "desired" response for the adaptive filter. The error signal is defined by

$$e(n) = d(n) - y(n) \tag{29}$$

Thus, substituting Eq. (22) in (28), we get

$$e(n) = s(n) + v_0(n) - y(n) \tag{30}$$

The error signal is, in turn, used to adjust the tap weights of the adaptive filter, and the control loop around the operations of filtering and subtraction is thereby closed. Note that the information-bearing signal $s(n)$ is indeed part of the error signal $e(n)$, as indicated in Eq. (30).

The error signal $e(n)$ constitutes the overall system output. From Eq. (30) we see that the noise component in the system output is $v_0(n)$ - $y(n)$. Now, the adaptive filter attempts to minimize the mean-square value (i.e., average power) of the error signal $e(n)$. The information-bearing signal $s(n)$ is essentially unaffected by the adaptive noise canceler. Hence, minimizing the mean-square value of the error signal $e(n)$ is equivalent to minimizing the mean-square value of the output noise $v_0(n)$ - $y(n)$. With the signal $s(n)$ remaining essentially constant, it follows that the minimization of the mean-square value of the error signal is indeed the same as the maximization of the output signal-to-noise ratio of the system.

The signal-processing operation described herein has two limiting cases that are noteworthy:

1. The adaptive filtering operation is perfect in the sense that

$$y(n) = v_0(n)$$

   In this case, the system output is noise free and the noise cancelation is perfect. Correspondingly, the output signal-to-noise ratio is infinitely large.

2. The reference signal $v_1(n)$ is completely uncorrelated with both the signal and noise components of the primary signal $d(n)$; that is,

P517567.PDF [Page: 37 of 78]

$$E[d(n)v_1(n-k)] = 0 \qquad \text{for all } k$$

In this case, the adaptive filter "switches itself off", resulting in a zero value for the output $y(n)$. Hence, the adaptive noise canceler has no effect on the primary signal $d(n)$, and the output signal-to-noise ratio remains unaltered.

The effective use of adaptive noise canceling therefore requires that we place the reference sensor in the noise field of the primary sensor with two specific objectives in mind. First, the information-bearing signal component of the primary sensor output is undetectable in the reference sensor output. Second, the reference sensor output is highly correlated with the noise component of the primary sensor output. Moreover, the adaptation of the adjustable filter coefficients must be near optimum.

In the remainder of this subsection, we describe three useful applications of the adaptive noise-canceling operation:

1. Canceling 60-Hz interference in electrocadiography. In electrocardiography (ECG), commonly used to monitor heart patients, an electrical discharge radiates energy through a human tissue and the resulting output is received by an electrode. The electrode is usually positioned in such a way that the received energy is maximized. Typically, however, the electrical discharge involves very low potentials. Correspondingly, the received energy is very small. Hence extra care has to be exercised in minimizing signal degradation due to external interference. By far, the strongest form of interference is that of a 60-Hz periodic waveform picked up by the receiving electrode (acting like an antenna) from nearby electrical equipment (Huhta and Webster, 1973). Needless to say, this interference has undesirable effects in the interpretation of electrocardiograms. Widrow et al. (1975b) have demonstrated the use of adaptive noise canceling (based on the LMS algorithm) as a method for reducing this form of interference. Specifically, the primary signal is taken from the ECG preamplifier, and the reference signal is taken from a wall outlet with proper attenuation. Figure 23 shows a block diagram of the adaptive noise canceler used by Widrow et al. (1975b). The adaptive filter has two adjustable weights, $\hat{w}_0(n)$ and $\hat{w}_1(n)$. One weight, $\hat{w}_0(n)$, is fed directly from the reference point. The second weight, $\hat{w}_1(n)$, is fed from a 90°-phase-shifted version of the reference input. The sum of the two weighted versions of the reference signal is then subtracted from the ECG output to produce an error signal. This error signal together with the weighted inputs are applied to the LMS algorithm, which, in turn, controls the adjustments applied to the two weights, In this application, the adaptive noise canceler acts as a variable "notch filter". The frequency of the sinusoidal interference in the ECG output is presumably the same as that of the sinusoidal reference signal. However, the amplitude and phase of the sinusoidal interference in the ECG output are unknown. The two weights, $\hat{w}_0(n)$ and, $\hat{w}_1(n)$ provide the two degrees of freedom required to control the amplitude and phase of the sinusoidal reference signal so as to cancel the 60-Hz interference contained in the ECG output.

2. Reduction of acoustic noise in speech. At a noisy site (e.g., the cockpit of a military aircraft), voice communication is affected by the presence of acoustic noise. This effect is particularly serious when linear predictive coding (LPC) is used for the digital representation of voice signals at low bit rates; LPC was discussed earlier. To be specific, high-frequency acoustic noise severely affects the estimated LPC spectrum in both the low- and high-frequency regions. Consequently,

P517567.PDF [Page: 38 of 78]

the intelligibility of digitized speech using LPC often falls below the minimum acceptable level. Kang and Fransen (1987) describe the use of an adaptive noise canceler, based on the LMS algorithm, for reducing acoustic noise in speech. The noise-corrupted speech is used as the primary signal. To provide the reference signal (noise only), a reference microphone is placed in a location where there is sufficient isolation from the source of speech (i.e., the known location of the speaker's mouth). In the experiments described by Kang and Fransen, a reduction of 10 to 15 dB in the acoustic noise floor is achieved, without degrading voice quality. Such a level of noise reduction is significant in improving voice quality, which may be unacceptable otherwise.

3. Adaptive speech enhancement. Consider the situation depicted in Fig. 24. The requirement is to listen to the voice of the desired speaker in the presence of background noise, which may be satisfied through the use of adaptive noise canceling. Specifically, reference microphones are added at locations far enough away from the desired speaker such that their outputs contain only noise. As indicated in Fig. 24, a weighted sum of the auxiliary microphone outputs is subtracted from the output of the desired speech-containing microphone, and an adaptive filtering algorithm (e.g., the LMS algorithm) is used to adjust the weights so as to minimize the average output power. A useful application of the idea described herein is in the adaptive noise cancelation for heating aids[6] (Chazan et al., 1988). The so-called "cocktail party effect" severely limits the usefulness of hearing aids. The cocktail party phenomenon refers to the ability of a person with normal hearing to focus on a conversation taking place at a distant location in a crowded room. This ability is lacking in a person who wears hearing aids, because of extreme sensitivity to the presence of background noise. This sensitivity is attributed to two factors: (a) the lose of directional cues, and (b) the limited channel capacity of the ear caused by the reduction in both dynamic range and frequency response. Chazan et al., (1988) describe an adaptive noise canceling technique aimed at overcoming this problem. The technique involves the use of an array of microphones that exploit the difference in spatial characteristics between the desired signal and the noise in a crowded room. The approach taken by Chazan et al. is based on the fact that each microphone output may be viewed as the sum of the signals produced by the individual speakers engaged in conversations in the room. Each signal contribution in a particular microphone output is essentially the result of a speaker's speech signal having passed through the room filter. In other words, each speaker (including the desired speaker) produces a signal at the microphone output that is the sum of the direct transmission of his or her speech signal and its reflections from the walls of the room. The requirement is to reconstruct the desired speaker signal, including its room reverberations, while canceling out the source of noise. In general, the transformation undergone by the speech signal from the desired speaker is not known. Also, the characteristics of the background noise are variable. We thus have a signal-processing problem for which adaptive noise canceling offers a feasible solution.

Adaptive Beamforming

For our last application, we describe a spatial form of adaptive signal processing that finds practical use in radar, sonar, communications, geophysical exploration, astrophysical exploration, and biomedical signal processing.

[6] This idea is similar to that of adaptive spatial filtering in the context of antennas, which is considered later in this section.

P517567.PDF [Page: 39 of 78]

In the particular type of spatial filtering of interest to us in this book, a number of independent sensors are placed at different points in space to "listen" to the received signal. In effect, the sensors provide a means of sampling the received signal in space. The set of sensor outputs collected at a particular instant of time constitutes a snapshot. Thus, a snapshot of data in spatial filtering (for the case when the sensors lie uniformly on a straight line) plays a role analogous to that of a set of consecutive tap inputs that exist in a transversal filter at a particular instant of time.[7]

In radar, the sensors consist of antenna elements (e.g., dipoles, horns, slotted waveguides) that respond to incident electromagnetic waves. In sonar, the sensors consist of hydrophones designed to respond to acoustic waves. In any event, spatial filtering known as beamforming, is used in these systems to distinguish between the spatial properties of signal and noise. The device used to do the beamforming is called a beamformer. The term "beamformer" is derived from the fact that the early forms of antennas (spatial filters) were designed to form pencil beams, so as to receive a signal radiating from a specific direction and attenuate signals radiating from other directions of no interest (Van Veen and Buckley, 1988). Note that he beamforming applies to the radiation (transmission) or reception of energy.

In a primitive type of spatial filtering, known as the delay-and-sum beamformer, the various sensor outputs are delayed (by appropriate amounts to align signal components coming from the direction of a target) and then summed, as in Fig. 25. Thus, for a single target, the average power at the output of the delay-and-sum beamformer is maximized when it is steered toward the target. A major limitation of the delay-and-sum beamformer, however, is that it has no provisions for dealing with sources of interference.

In order to enable a beamformer to respond to an unknown interference environment, it has to be made adaptive in such a way that it places nulls in the direction(s) of the source(s) of interference automatically and in real time. By so doing, the output signal-to-noise ratio of the system is increased, and the directional response of the system is thereby improved. Below, we consider two examples of adaptive beamformers that are well suited for use with narrow-band signals in radar and sonar systems.

Adaptive beamformer with minimum-variance distortionless response. Consider an adaptive beamformer that uses a linear array of M identical sensors, as in Fig. 26. The individual sensor outputs, assumed to be in baseband form are weighted and then summed. The beamformer has to satisfy two requirements: (1) a steering capability whereby the target signal is always protected, and (2) the effects of sources of interference are minimized. One method of providing for these two requirements is to minimize the variance (i.e., average power) of the beamformer output, subject to the constraint that, during the process of adaptation, the weights satisfy the condition:

$$\mathbf{w}^H(n)\mathbf{s}(\phi) = 1 \qquad \text{for all } n \text{ and } \phi = \phi_t \tag{31}$$

---

[7] For a discussion of the analogies between time- and space-domain forms of signal processing, see Bracewell (1986) and Van Veen and Buckley (1988).

P517567.PDF [Page: 40 of 78]

where $\mathbf{w}(n)$ is the M-by-1 weight vector and $\mathbf{s}(\phi)$ is an M-by-1 steering vector. The superscript H denotes Hermitian transposition (i.e., transposition combined with complex conjugation). In this application, the baseband data are complex valued; hence the need for complex conjugation. The value of electrical angle $\phi = \phi_t$ is determined by the direction of the target. The angle ϕ is itself measured with sensor 1 (at the top end of the array) treated as the point of reference.

The dependence of vector $\mathbf{s}(\phi)$ on the angle ϕ is defined by

$$\mathbf{s}(\phi) = [1, e^{-j\phi}, \cdots, e^{-j(M-1)\phi}]^T$$

The angle ϕ is itself related to incidence angle θ of a plane wave, measured with respect to the normal to the linear array, as follows[8]

$$\phi = \frac{2\pi d}{\lambda} \sin\theta \tag{32}$$

where d is the spacing between adjacent sensors of the array, and λ is the wavelength (see Fig. 27). The incidence angle θ lies inside the range $-\pi/2$ to $\pi/2$. The permissible values that the angle ϕ may assume lie inside the range $-\pi$ to π. This means that we must choose the spacing $d < \lambda/2$, so that there is a one-to-one correspondence between the values of θ and ϕ without ambiguity. The condition $d < \lambda/2$ may be viewed as the spatial analog of the sampling theorem.

The imposition of the signal-protection constraint in Eq. (31) ensures that, for a prescribed look direction, the response of the array is maintained constant (i.e., equal to 1), no matter what values are assigned to the weights. An algorithm that minimizes the variance of the beamformer output, subject to this constraint, is therefore referred to as the minimum-variance distortionless response (MVDR) beamforming algorithm (Capon, 1969; Owsley, 1986). The imposition of the constraint described in Eq. (31) reduces the number of "degrees of freedom" available to the MVDR algorithm to $M - 2$, where M is the number of sensors in the array. This means that the number of independent nulls produced by the MVDR algorithm (i.e., the number of independent interferences that can be canceled is $M - 2$.

The MVDR beamforming is a special case of linearly constrained minimum variance (LCMV) beamforming. In the latter case, we minimize the variance of the beamformer output, subject to the constraint

$$\mathbf{w}^H(n)\mathbf{s}(\phi) = g \quad \text{for all } n \text{ and } \phi = \phi_t \tag{33}$$

where g is a complex constant. The LCMV beamformer linearly constrains the weights, such that any signal coming from electrical angle ϕ_t is passed to the output with response (gain) g.

---

[8] When a plane wave impinges on a linear array as in Fig. 27 there is a spatial delay of $d \sin\theta$ between the signals received at any pair of adjacent sensors. With a wavelength of λ, this spatial delay is translated into an electrical angular difference defined by $\phi = 2\pi(d \sin\theta/\lambda)$.

P517567.PDF [Page: 41 of 78]

Comparing the constraint of Eq. (31) with that of Eq. (33), we see that the MVDR beamformer is indeed a special case of the LCMV beamformer for $g = 1$.

Adaptation in beam space. The MVDR beamformer performs adaptation directly in the data space. The adaptation process for interference cancelation may also be performed in beam space. To do so, the input data (received by the array of sensors) are transformed into the beam space by means of an orthogonal multiple-beamforming network, as illustrated in the block diagram of Fig. 28. The resulting output is processed by a multiple sidelobe canceler so as to cancel interference(s) from unknown directions.

The beamforming network is designed to generate a set of orthogonal beams. The multiple outputs of the beamforming network are referred to as beam ports. Assume that the sensor outputs are equally weighted and have a uniform phase. Under this condition, the response of the array produced by an incident plane wave arriving at the array along direction θ, measured with respect to the normal to the array, is given by

$$A(\phi, \alpha) = \sum_{n=-N}^{N} e^{jn\phi} e^{-jn\alpha} \tag{34}$$

where $M = (2N + 1)$ is the total number of sensors in the array, with the sensor at the midpoint of the array treated as the point of reference. The electrical angle ϕ is related to θ by Eq. (32), and α is a constant called the uniform phase factor. The quantity $A(\phi,\alpha)$ is called the array pattern. For $\phi = \lambda/2$, we find from Eq. (31) that

$$\phi = \pi \sin\theta$$

Summing the geometric series in Eq. (34), we may express the array pattern as

$$A(\phi, \alpha) = \frac{\sin\left[\frac{1}{2}(2N + 1)(\phi - \alpha)\right]}{\sin\left[\frac{1}{2}(\phi - \alpha)\right]} \tag{35}$$

By assigning different values to α, the main beam of the antenna is thus scanned across the range $-\pi < \phi \leq \pi$. To generate an orthogonal set of beams, equal to $2N$ in number, we assign the following discrete values to the uniform phase factor

$$\alpha = \frac{\pi}{2N + 1} k, \qquad k = \pm 1, \pm 3, \cdots, \pm 2N - 1 \tag{36}$$

Figure 29 illustrates the variations of the magnitude of the array pattern $A(\phi,\alpha)$ with ϕ for the case of $2N + 1 = 5$ elements and $\alpha = \pm\pi/5, \pm 3\pi/5$. Note that owing to the symmetric nature of the beamformer, the final values of the weights are real valued.

P517567.PDF [Page: 42 of 78]

The orthogonal beams generated by the beamforming network represent $2N$ independent look directions, one per beam. Depending on the target direction of interest, a particular beam in the set is identified as the main beam and the remainder are viewed as auxiliary beams. We note from Fig. 29 that each of the auxiliary beams has a null in the look direction of the main beam. The auxiliary beams are adaptively weighted by the multiple sidelobe canceler so as to form a cancelation beam that is subtracted from the main beam. The resulting estimation error is fed back to the multiple sidelobe canceler so as to control the corrections applied to its adjustable weights.

Since all the auxiliary beams have nulls in the look direction of the main beam, and the main beam is excluded from the multiple sidelobe canceler, the overall output of the adaptive beamformer is constrained to have a constant response in the look direction of the main beam (i.e., along the direction of the target). Moreover, with $(2N - 1)$ degrees of freedom (i.e., the number of available auxiliary beams) the system is capable of placing up to $(2N - 1)$ nulls along the (unknown) directions of independent interferences.

Note that with any array of $(2N + 1)$ sensors, we may produce a beamforming network with $(2N + 1)$ orthogonal beam ports by assigning the uniform phase factor the following set of values:

$$\alpha = \frac{k\pi}{2N-1} \qquad k = 0, \pm 2, \cdots, \pm 2N \tag{37}$$

In this case, a small fraction of the main lobe of the beam port at either end lies in the nonvisible region. Nevertheless, with one of the beam ports providing the main beam and the remaining $2N$ ports providing the auxiliary beams, the adaptive beamformer is now capable of producing up to $2N$ independent nulls.

8. Concluding Remarks

Adaptive signal processing, be it in time, space, or space-time, is essential to the design of modern radar, sonar, and biomedical imaging systems. We say so for the following reasons:

- The underlying statistics of the signals of interest may be unknown, which makes it difficult, if not impossible, to design optimum filters by classical methods. An adaptive signal processor overcomes this difficulty by learning the underlying statistics of the environment in an on-line fashion, off-line fashion, or combination thereof.

- Signals generated by radar, sonar, and biomedical systems are inherently nonstationary. Adaptive signal processing provides an elegant approach to deal with nonstationary phenomena by adjusting the free parameters of a filter in accordance with prescribed algorithms.

In this chapter, we have presented a guided tour of Adaptive Systems for Signal Processing by focusing attention on the following issues:

- Linear adaptive systems, exemplified by the LMS and RLS algorithms. The LMS algorithm is simple to design but slow to converge. The RLS algorithm, in contrast, is complex but fast to

P517567.PDF [Page: 43 of 78]

converge. When the adaptive filtering is of a temporal kind, the complexity of the RLS algorithm can be reduced significantly by exploiting the time-shifting property of the input signals; for details, see Haykin (1996).

- Nonlinear adaptive systems, exemplified by neural networks. This class of adaptive systems takes many different forms. The network can be of the feedforward type, which is exemplified by multilayer perceptrons and radial-basis function networks. For the design of multilayer perceptrons we can use the back-propagation algorithm, which is a generalization of the LMS algorithm. A more principled approach for the design of multilayer perceptrons and radial-basis function networks is to use support vector machine theory pioneered by Vapnik and co-workers (Vapnik, 1998). Another important type of neural networks involves the abundant use of feedback, which is exemplified by Hopfield networks and dynamically driven recurrent networks. The state-space approach of modern control theory provides a powerful basis for the design of dynamically driven recurrent networks, so as to synthesize input-output mappings of interest.

The choice of one of these adaptive systems over another can only be determined by the application of interest.

To motivate the study of these two broadly defined classes of adaptive systems, we described five different applications:

- system identification
- spectrum estimation
- noise cancellation
- signal detection
- beamforming

The need for each and everyone of these applications arises in radar, sonar, and medical imaging systems in a variety of different ways, as illustrated in the subsequent chapters of the book.

P517567.PDF [Page: 44 of 78]

References

Åström, K.J., and P. Eykhoff, (1971). "System identification--a survey", Automatica, vol. 7, pp. 123-162.

Barlow, H.B. ,1(989). "Unsupervised learning", Neural Computation, vol. 1, pp. 295-311.

Bar-Shalom, Y., and T.E. Fortmann, (1988). Tracking and Data association, Academic Press.

Becker, S., and G.E. Hinton, (1992). "A self-organizing neural network that discovers surfaces in random-dot stereograms", Nature (London), vol. 355, pp. 161-163.

Bengio, Y., P. Simard, and P. Frasconi, (1994). "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, vol. 5, pp. 157-166.

Bracewell, R.N., (1978.) The Fourier Transform and Its Applications, 2nd ed., McGraw-Hill, New York.

Broomhead, D.S., and D. Lowe, (1988). "Multivariable functional interpolation and adaptive networks", Complex Systems, vol. 2, pp. 321-355.

Capon, J., (1969). "High-resolution frequency-wavenumber spectrum analysis", Proc. IEEE, vol. 57, pp. 1408-1418.

Chazan, D., Y. Medan, and U. Shvadron, (1988). "Noise cancellation for hearing aids", IEEE Trans. Acoust. Speech Signal Process., vol. ASSP-36, pp. 1697-1705.

Cohen, L., 1(995). Time-Frequency Analysis, Prentice-Hall, Englewood-Cliffs, NJ.

Comon, P., (1994). "Independent component analysis: A new concept?" Signal Processing, vol. 36, pp. 287-314.

El Hihi, S., and Y. Bengio, (1996). "Hierarchical recurrent neural networks for long-term dependencies", Advances in Neural Information Processing Systems, vol. 8, pp. 493-499, MIT Press.

Feldkamp, L.A., and G.V. Puskorius, (1998). "A signal processing framework based on dynamic neural networks with application to problems in adaptation, filtering and classification", Special issue of Proceedings of the IEEE on Intelligent Signal Processing, vol. 86, November.

Goodwin, G.C., and R.L. Rayne, (1977). Dynamic System Identification: Experiment Design and Data Analysis, Academic Press, New York.

Haykin, S., (1996). Adaptive Filter Theory, 3rd Edition, Prentice-Hall.

Haykin, S., (1999). Neural Networks: A Comprehensive Foundation, 2nd Edition, Prentice-Hall.

P517567.PDF [Page: 45 of 78]

Haykin, S., and T. Bhattacharya, (1997). "Modular learning strategy for signal detection in a nonstationary environment", IEEE Trans. Signal Processing, vol. 45, pp. 1619-1637.

Haykin, S., and D. Thomson (1998), "Signal Detection in a Nonstationary Environment Reformulated as an Adaptive Pattern Classification Problem." Procc IEEE Vol. 86, No. 11, pp. 2325-2344.

Hochreiter, S., and J. Schmidhuber, (1997). "LSTM can solve hard long time lag problems", Advances in Neural Information Processing Systems, vol. 9, pp. 473-479, Cambridge, MA: MIT Press.

Itakura, F. and S. Saito, (1972). "On the optimum quantization of feature parameters in the PARCOR speech synthesizer", in IEEE 1972 Conf. Speech Commun. Process., New York, pp. 434-437.

Johnson, D.H., and P.S. Rao, (1990). "On the existence of Gaussian noise", in The 1990 Digital Signal Processing Workshop, New Paltz, NY, Sponsored by IEEE Signal Processing Society, pp. 8.13.1-8.14.2.

Julier, J.J., and J.K. Uhlmann, (1997). "A New Extension of the Kalman Filter to Nonlinear Systems". In Proc. Eleventh International Symposium on Aerospace/Defence Sensing, Simulation, and Controls, Orlando, Florida.

Kallmann, H.J., (1940). "Transversal filters", Proc. IRE, vol. 28, pp. 302-310.

Kang G.S., and L.J. Fransen, (1987). "Experimentation with an adaptive noise-cancellation filter", IEEE Trans. Circuits Syst., vol. CAS-34, pp. 753-758.

Kohonen, T. (1997). Self-Organizing Maps, 2nd edition, Springer Verlag.

Kung, H.T., and C.E. Leiserson, (1978). "Systolic arrays (for VLSI)", Sparse Matrix Proc. 1978, Soc. Ind. Appl. Math., 1978, pp. 256-282.

Ljung, L., and T. Söderström, 1983. Theory and Practice of Recursive Identification, MIT Press, Cambridge, Mass.

Linsker, R., (1988). Towards an organizing principle for a layered perceptual network", in Neural Information Processing Systems, D.Z. Anderson, ed., pp. 485-494, New York: American Institute of Physics.

R. Linsker, (1989). "How to generate ordered maps by maximizing the mutual information between input and output signals", Neural Computation, vol. 1, pp. 402-411.

Luenberger, D.G., (1969). "Optimization by Vector Space Methods, Wiley, New York.C.N. Dorny, 1975. A Vector Space Approach to Models and Optimization, Wiley-Interscience, New York.

P517567.PDF [Page: 46 of 78]

Luttrell, S.P. (1989). "Self-organization: A derivation from first principle of a class of learning algorithms", IEEE Conference on Neural Networks, pp. 495-498, Washington, D.C.

Moody and Darken, (1989). "Fast learning in networks of locally-tuned processing units", Neural Computation, vol. 1, pp. 281-294.

Oja, E., (1982). "A simplified neuron model as a principal component analyzer", Journal of Mathematical Biology, vol. 15, pp, 267-273.

Owsley, N.L. (1985). "Sonary array processing", in Array Signal Processing, ed. S. Haykin, Prentice Hall, Englewood Cliffs, NJ, pp. 115-193.

Poggio, T., and F. Girosi, (1990). "Networks for approximation and learning", Proceedings of the IEEE, vol. 78, pp. 1481-1497.

Rumelhart, D.E., G.E. Hinton, and R.J. Williams, (1986). "Learning internal representations by error propagation", in D.E. Rumelhart and J.L. McCleland, eds. vol. 1, Chapter 8, Cambridge, MA: MIT Press.

Sanger, T.D., (1989). "An optimality principle for unsupervised learning", Advances in Neural Information Processing Systems, vol. 1, pp. 11-19, San Mateo, CA: Morgan Kaufmann.

Shabhag, N.R., and K.K. Parhi, (1994). Pipelined Adaptive Digital Filters, Kluwer, Boston, Mass.

Sontag, E.D., (1990). Mathematical Control Theory: Deterministic Finite Dimensional Systems, Springer-Verlag, New York.

Ukrainec, A.M., and S. Haykin, (1996). "A modular neural network for enhancement of cross-polar radar targets", Neural Networks, vol. 9, pp. 143-168.

Van Trees, H.L. (1968). Detection, Estimation and Modulation Theory, part I, Wiley, New York.

Van Veen, B.D., and K.M. Buckley, (1988). "Beamforming: a versatile approach to spatial filtering", IEEE ASSP Mag., vol. 5, pp. 4-24.

Vapnik, V.N., (1998). Statistical Learning Theory, New York: Wiley.

Wan, E.A., (1994). "Time series prediction by using a connectionist network with internal delay lines", in Time Series Prediction: Forecasting the Future and Understanding the Past, A.S. Weigend and N.A. Gershenfield, eds., pp. 195-217. Reading, MA: Addison-Wesley.

Wan, E.A., R. vander Merwe and A.T. Nelson, (1999). "Dual Estimation and the Unsented Transformation", Neural Information Processing Systems (NIPS),Denver, Colorado.

P517567.PDF [Page: 47 of 78]

Werbos, P.J., (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences, Ph.D. Thesis, Harvard University, Cambridge, MA.

Werbos, P.J., (1990). "Backpropagation through time: What it does and how to do it", Proceedings of the IEEE, vol. 78, pp. 1550-1560.

Widrow, B., et. al. (1975b). "Adaptive noise cancelling: principles and applications", Proc. IEEE, vol. 63, pp. 1692-1716.

Widrow B., and S.D. Stearns, (1985). Adaptive Signal Processing, Prentice-Hall, Englewood Cliffs, NJ.

Williams, R.J., and D. Zipser, (1989). A learning algorithm for continually running fully recurrent neural networks", Neural Computation, vol. 1, pp. 270-280.

Yee, P.V., 1998. Regularized Radial Basis Function Networks: Theory and Applications to Probability Estimation, Classification, and Time Series Prediction, Ph.D. Thesis, McMaster University, Hamilton, Ontario.

P517567.PDF [Page: 48 of 78]

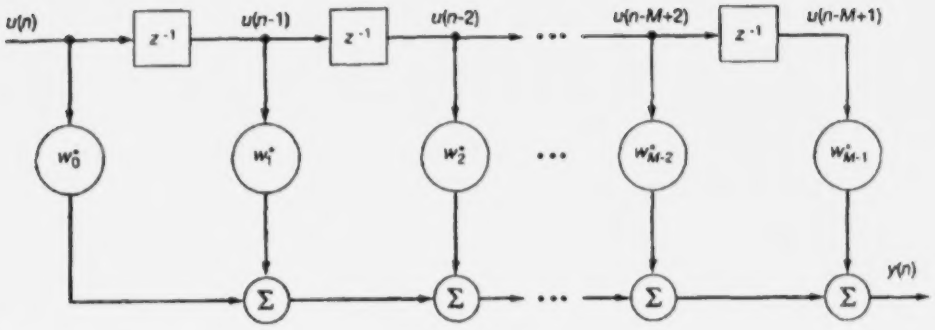

Figure 1: Transversal filter

P517567.PDF [Page: 49 of 78]

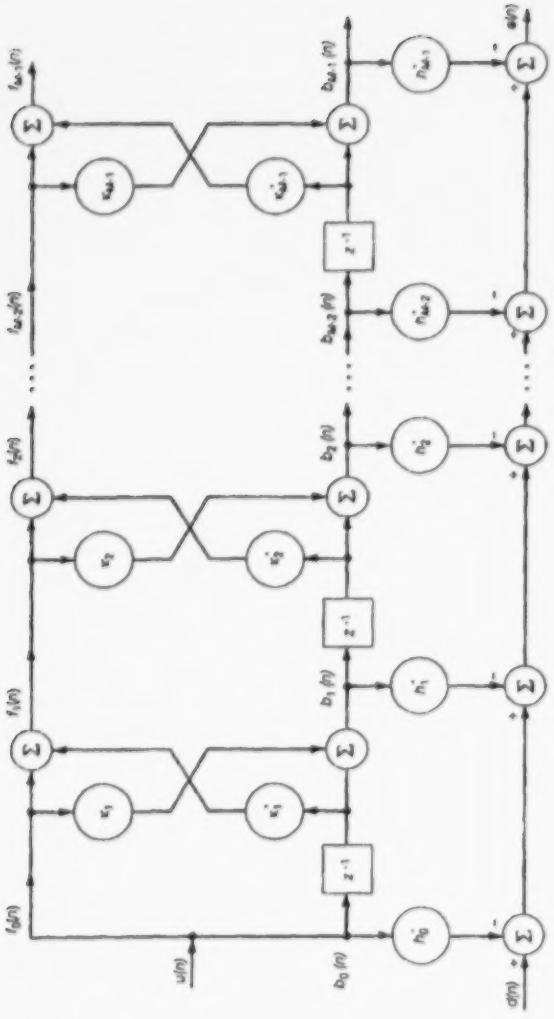

Figure 2: Multistage lattice filter

P517567.PDF [Page: 50 of 78]

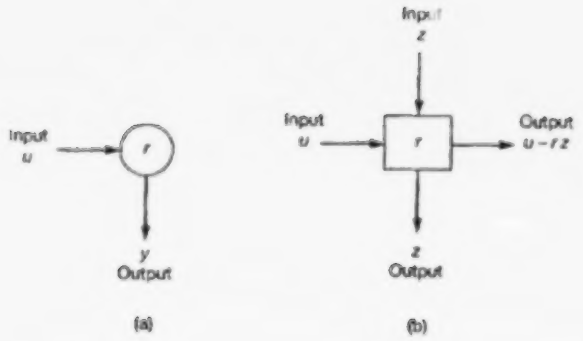

Figure 3: Two basic cells of a systolic array; (a) boundary cell; (b) internal cell

P517567.PDF [Page: 51 of 78]

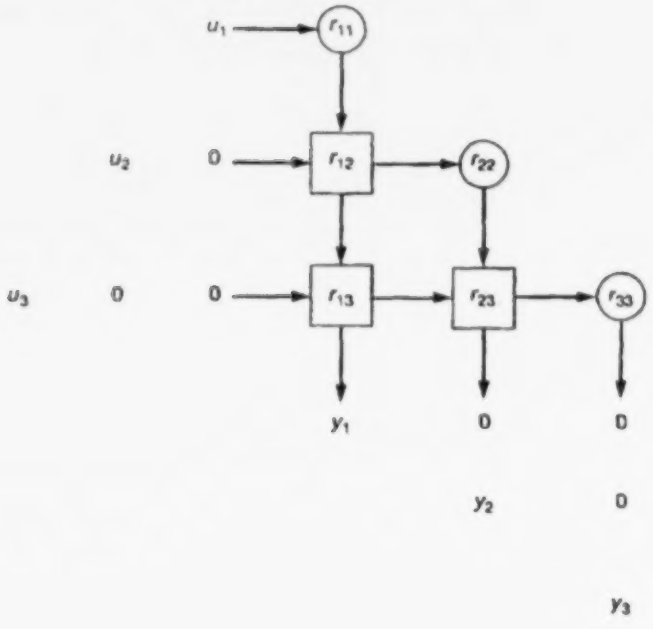

Figure 4: Triangular systolic array

P517567.PDF [Page: 52 of 78]

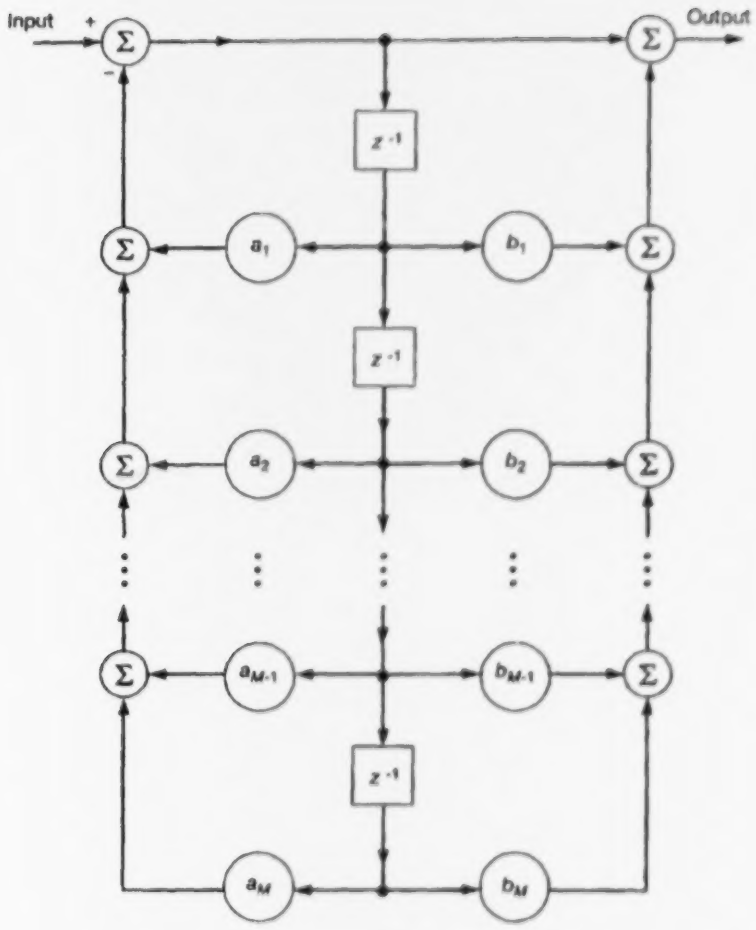

Figure 5: IIR filter

P517567 PDF [Page: 53 of 78]

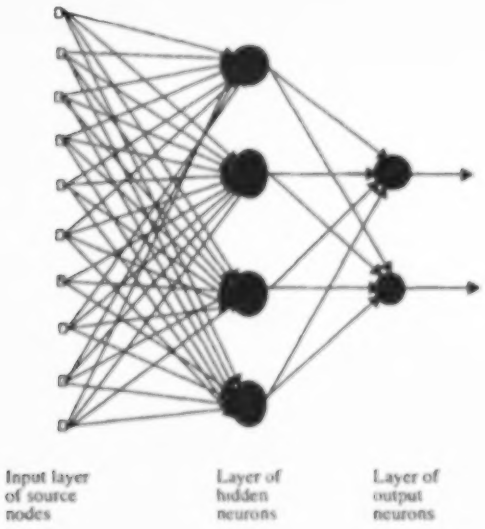

Figure 6: Fully connected feedforward of acyclic network with one hidden layer and one output layer

P517567.PDF [Page: 54 of 78]

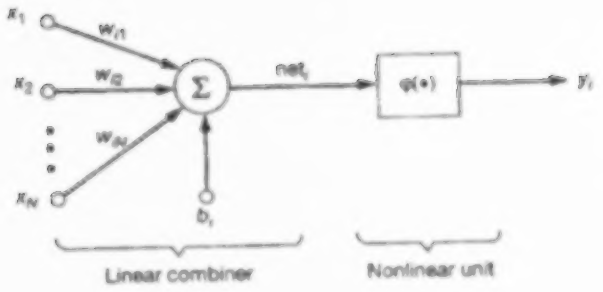

Figure 7: Simplified model of a neuron

P517567.PDF [Page: 55 of 78]

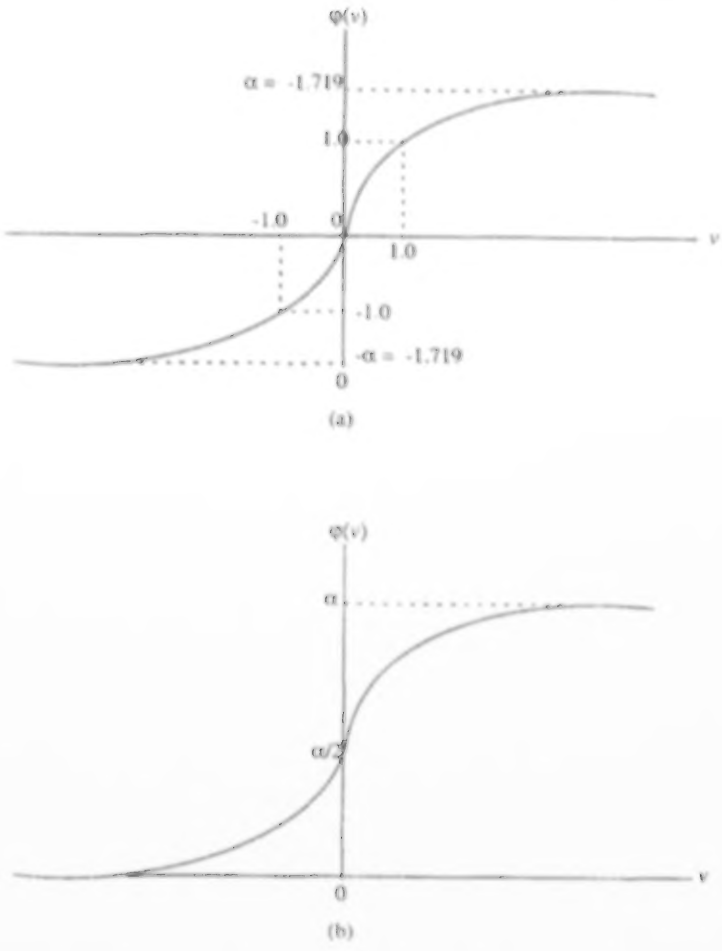

Figure 8: Antisymmetric activation function. (b) Nonsymmetric activation function.

P517567.PDF [Page: 56 of 78]

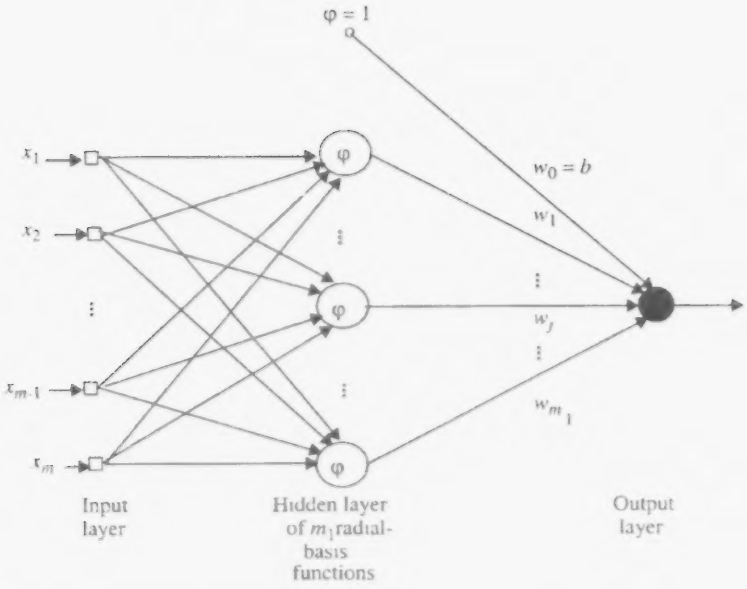

Figure 9: Radial-basis function network

P517567.PDF [Page: 57 of 78]

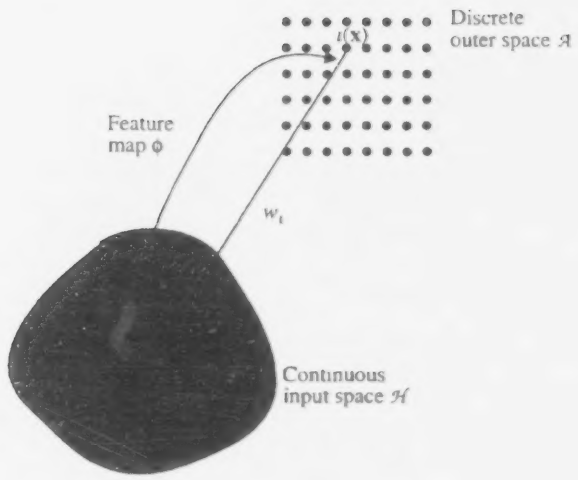

Figure 10: Illustration of the relationship between feature map ϕ and weight vector $\mathbf{w}_i$ of winning neuron i.

P517567.PDF [Page: 58 of 78]

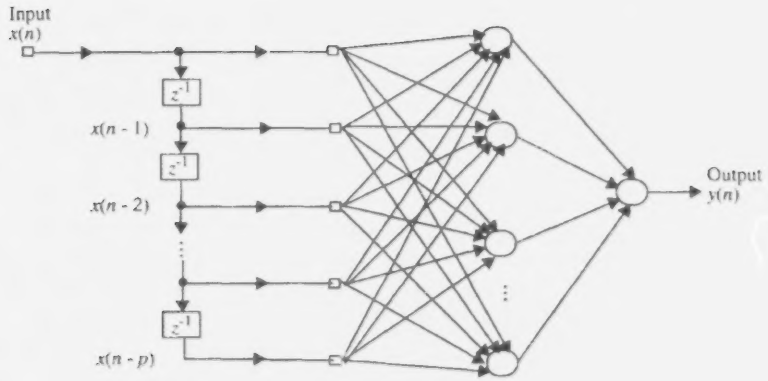

Figure 11: Focused time lagged feedforward network (TLFN); the bias levels have been omitted for convenience of presentation

P517567.PDF [Page: 59 of 78]

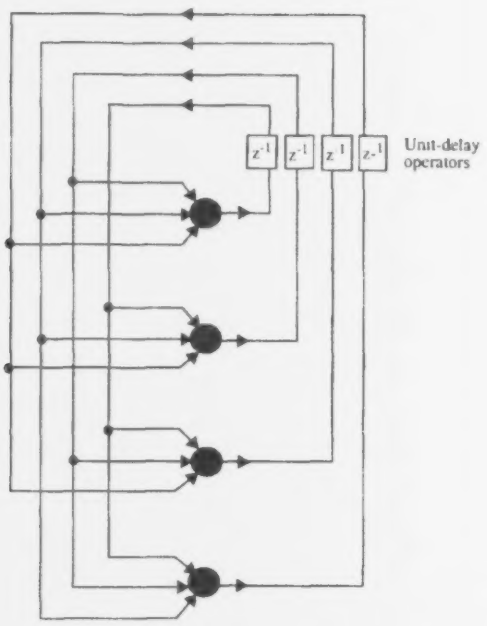

Figure 12: Recurrent network with no self-feedback loops and no hidden neurons

P517567.PDF [Page: 60 of 78]

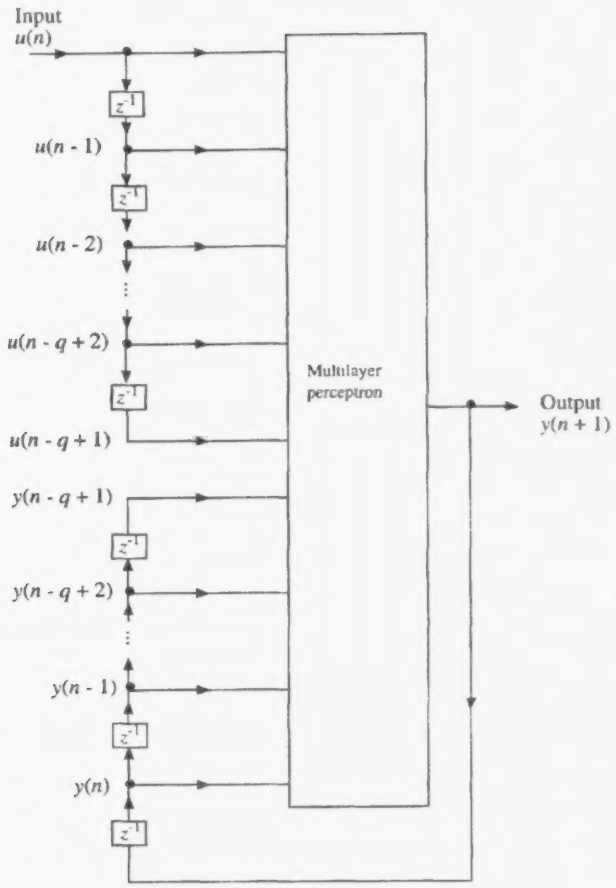

Figure 13: Nonlinear autoregressive with exogenous inputs (NARX) model

P517567.PDF [Page: 61 of 78]

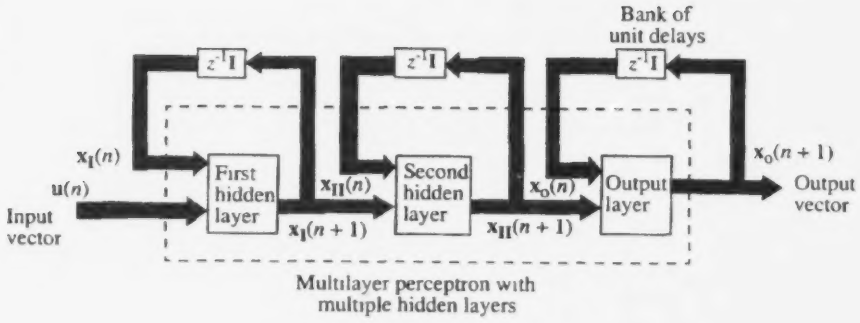

Figure 14: Recurrent multilayer perceptron

P517567.PDF [Page: 62 of 78]

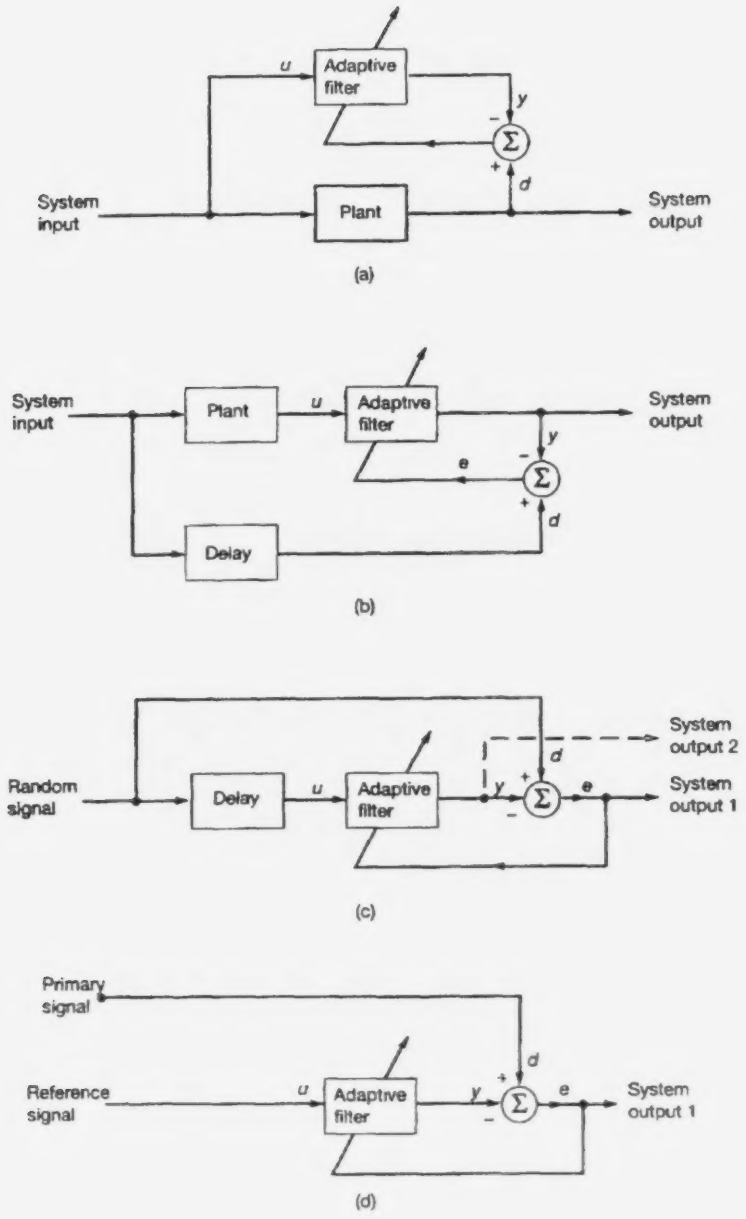

Figure 15: Four basic classes of adaptive filtering applications: (a) class I: identification; (b) class II: inverse modeling; (c) class III: prediction; (d) class IV: interference canceling

P517567 PDF [Page: 63 of 78]

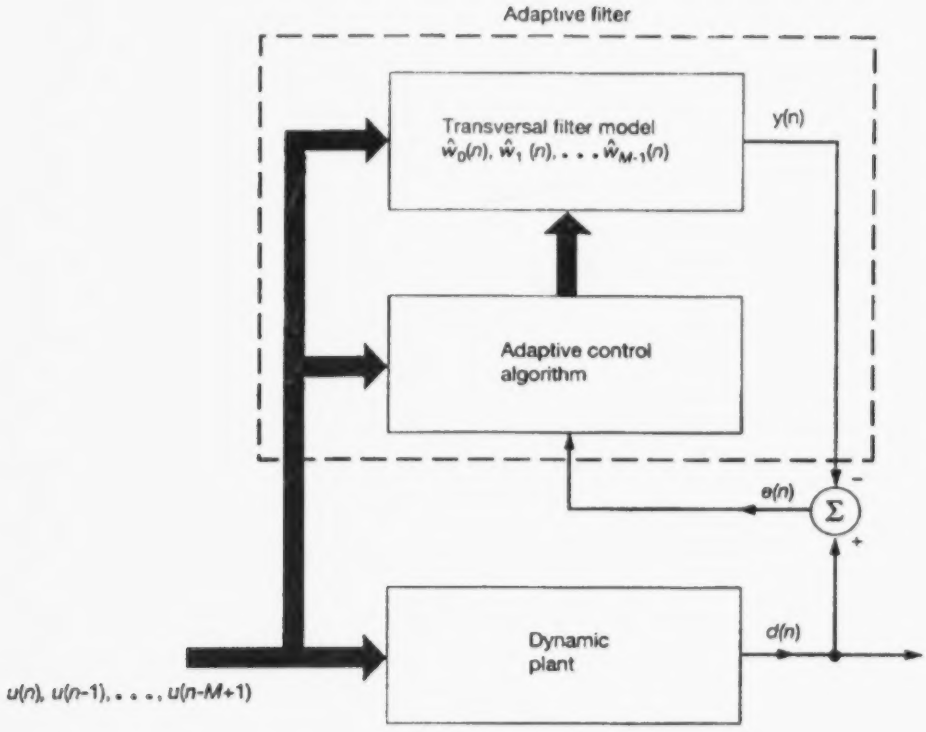

Figure 16: System identification

P517567.PDF [Page: 64 of 78]

Figure 17: Black box representation of a stochastic model

P517567.PDF [Page: 65 of 78]

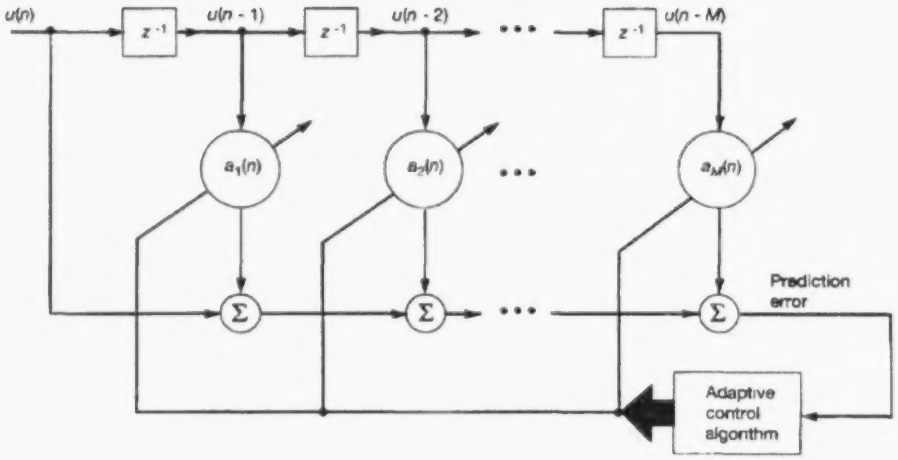

Figure 18: Adaptive prediction-error filter for real-valued data

P517567.PDF [Page: 66 of 78]

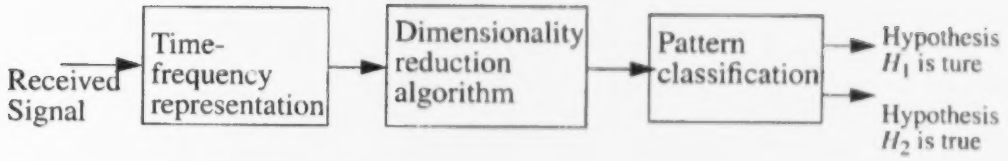

Figure 19:

P517567.PDF [Page: 67 of 78]

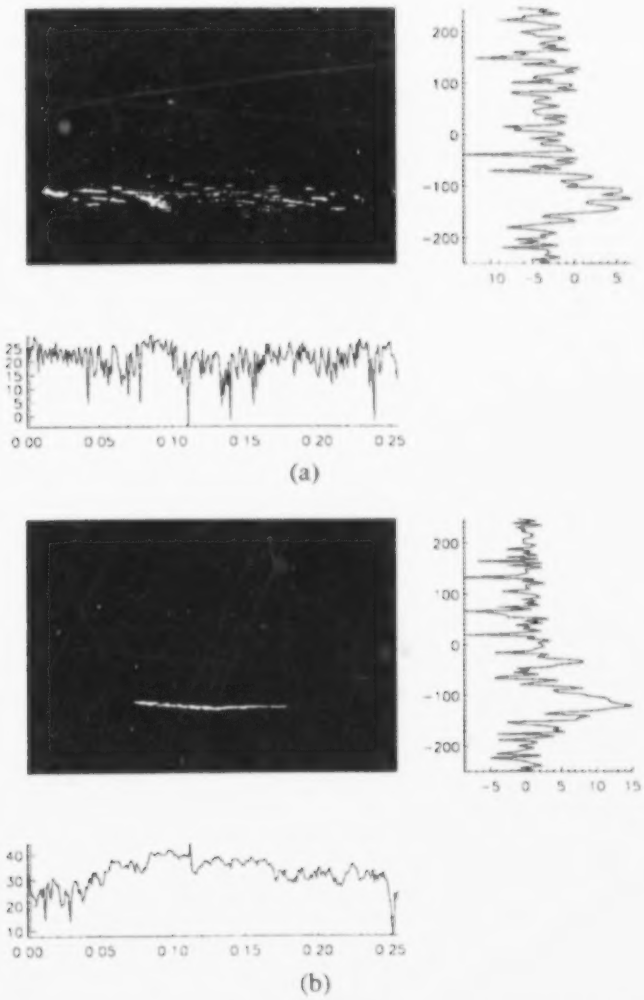

Figure 20: (a) WVD for sea clutter. (b) WVD for a barely visible growler. For the images, horizontal axes: time in seconds; vertical axes: frequency in Hz. Horizontal axes of power spectra: in dB.

P517567 PDF [Page: 68 of 78]

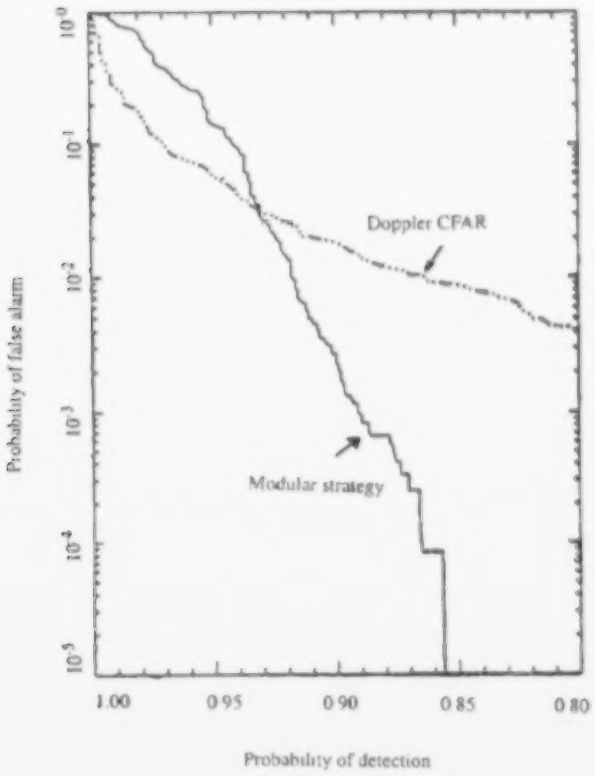

Figure 21: Composite Receiver Operating Characteristics

P517567.PDF [Page: 69 of 78]

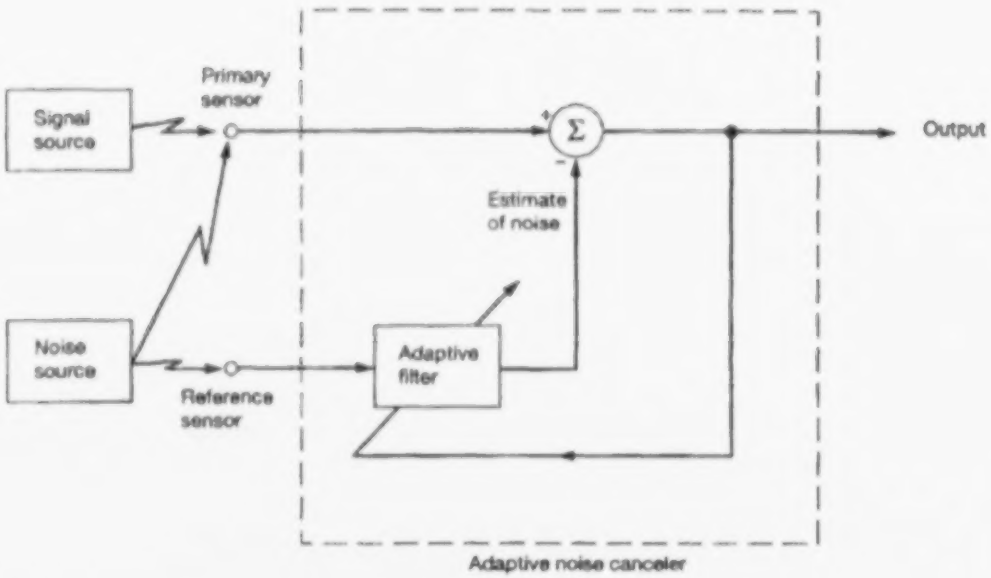

Figure 22: Adaptive noise cancelation

P517567 PDF [Page: 70 of 78]

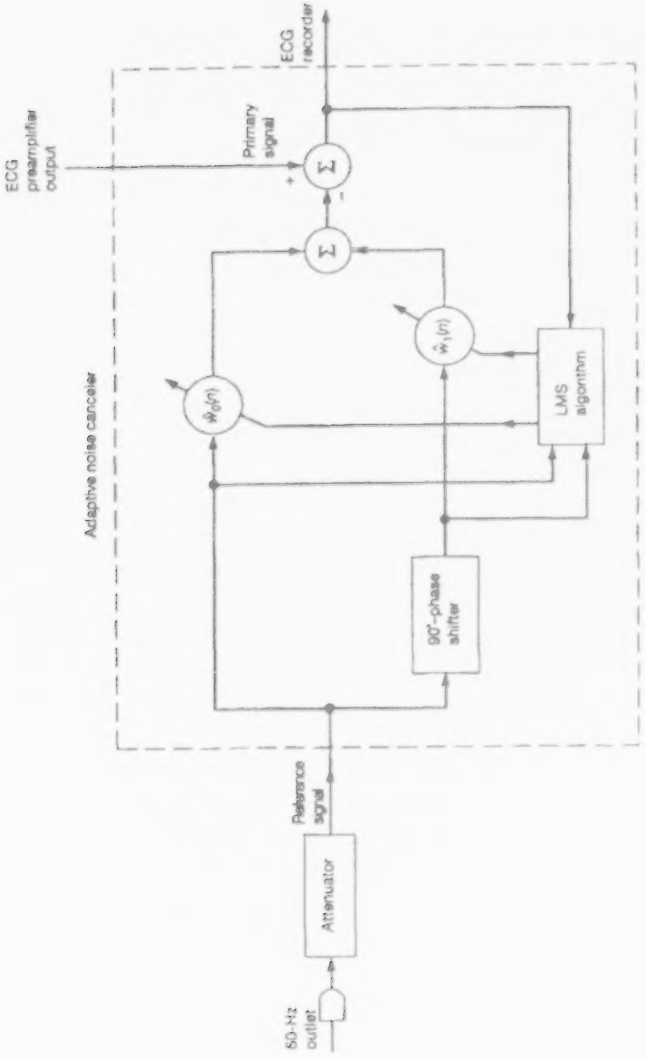

Figure 23: Adaptive noise canceler for suppressing 60-Hz interference in electrocardiography (After Widrow et al., 1975b)

P517567.PDF [Page: 71 of 78]

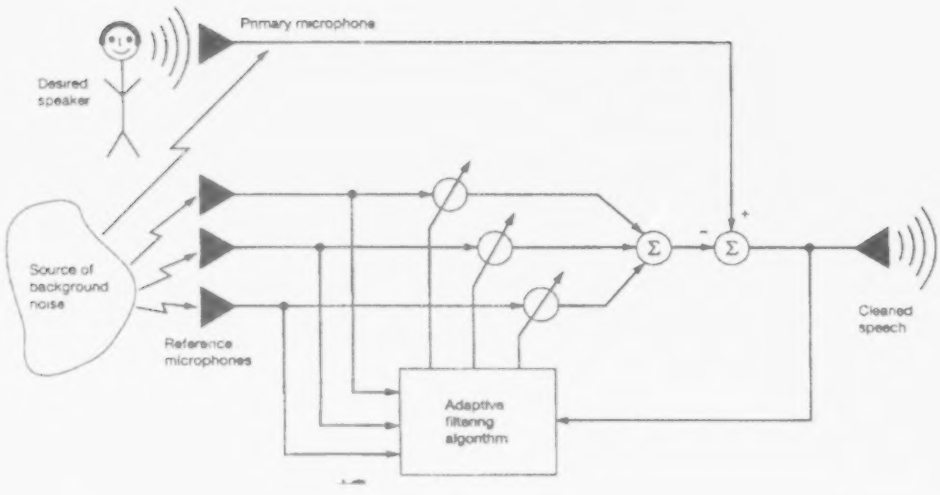

Figure 24: Block diagram of an adaptive noise canceler for speech.

P517567.PDF [Page: 72 of 78]

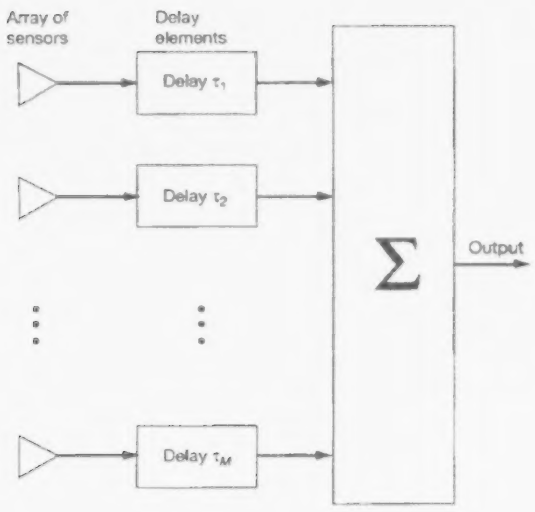

Figure 25: Delay-and-sum beamformer

P517567 PDF [Page: 73 of 78]

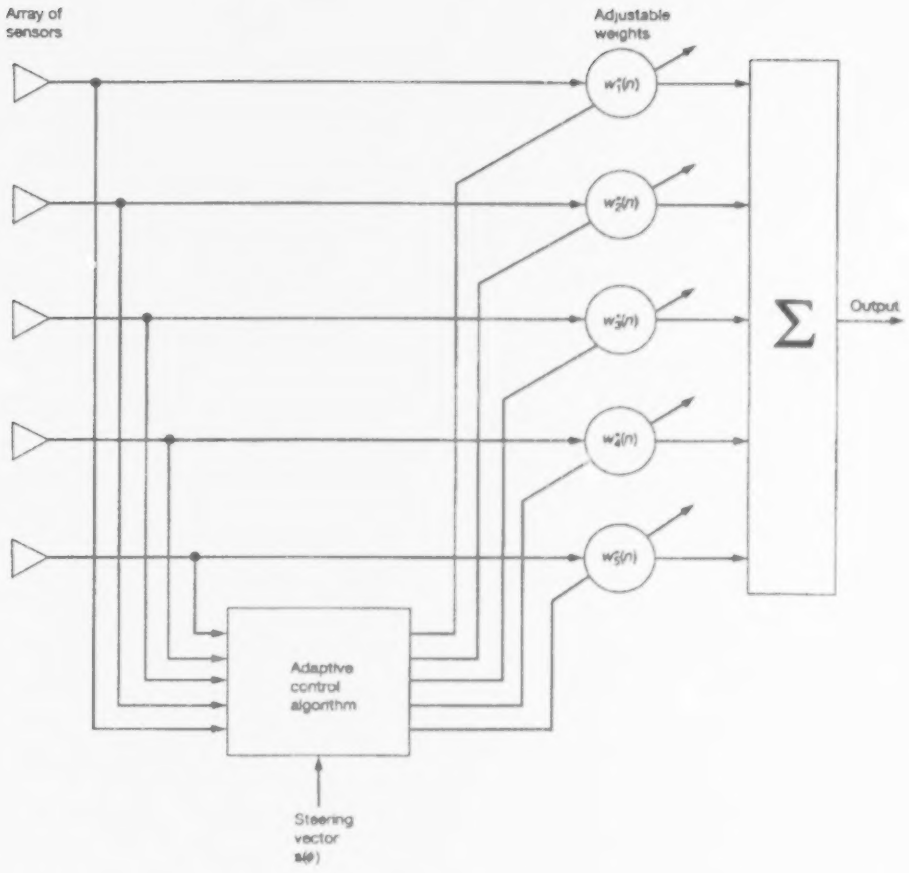

Figure 26: Adaptive beamformer for an array of 5 sensors. The sensor outputs (in baseband form) are complex valued; hence the weights are complex valued

P517567.PDF [Page: 74 of 78]

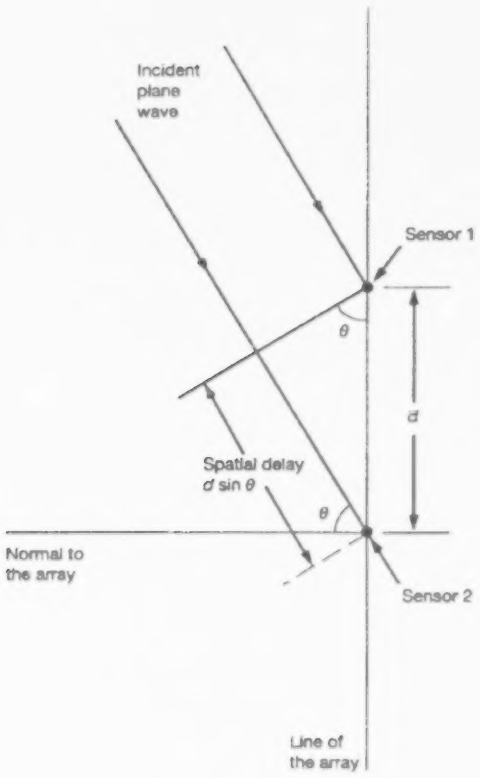

Figure 27: Spatial delay incurred when a plane wave impinges on a linear array

P517567 PDF [Page: 75 of 78]

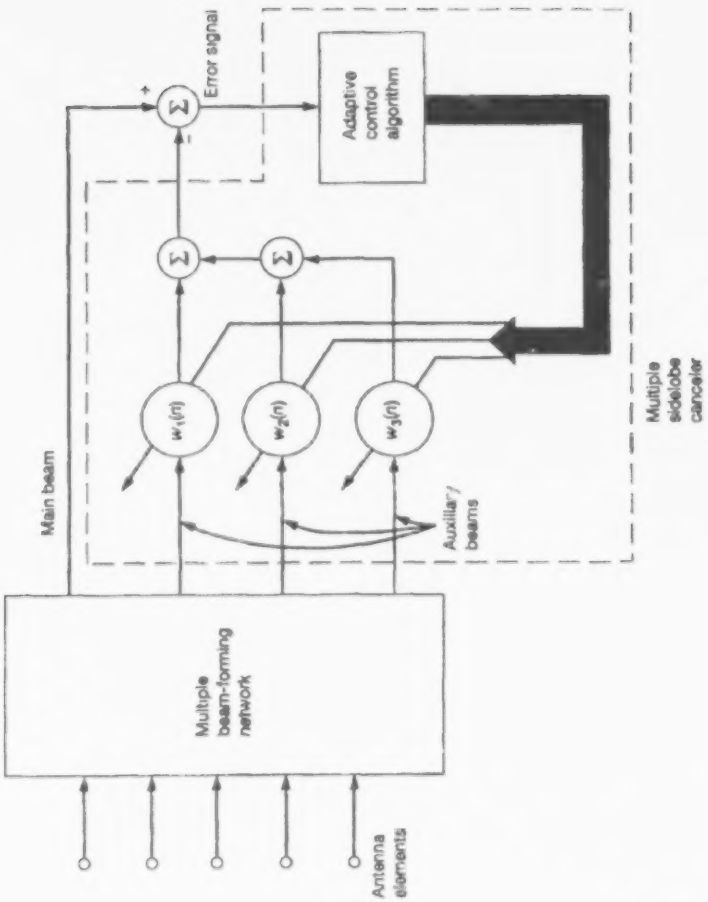

Figure 28: Block diagram of adaptive combiner with fixed beams; owing to the symmetric nature of the multiple beamforming network, final values of the weights are real valued

P517567.PDF [Page: 76 of 78]

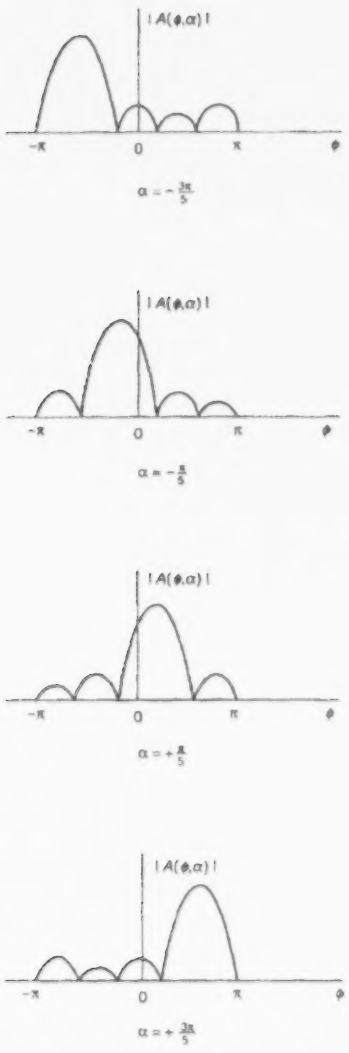

Figure 29: Variations of the magnitude of the array pattern $A(\phi,\alpha)$ with ϕ and α

DOCUMENT CONTROL DATA SHEET

1a PERFORMING AGENCY	2 SECURITY CLASSIFICATION
McMaster University, Hamilton, ON	UNCLASSIFIED Unlimited distribution -
1b PUBLISHING AGENCY	
DCIEM	

3 TITLE

(U) Adaptive systems for signal processing

4 AUTHORS

Simon Haykin

5 DATE OF PUBLICATION	6 NO OF PAGES
March 1 , 2002	73

7 DESCRIPTIVE NOTES

8 SPONSORING/MONITORING/CONTRACTING/TASKING AGENCY

Sponsoring Agency

Monitoring Agency

Contracting Agency DCIEM

Tasking Agency

9 ORIGINATORS DOCUMENT NO	10 CONTRACT GRANT AND/OR PROJECT NO	11 OTHER DOCUMENT NOS
Contract Report CR 2002-045	W7711-9-7589/001/TOR	

12 DOCUMENT RELEASABILITY

Unlimited distribution

13. DOCUMENT ANNOUNCEMENT

Unlimited announcement

P517567.PDF [Page: 78 of 78]

14 ABSTRACT

(U) The present report discusses in some detail the recent developments in adaptive signal processing and their advantages for real-time system applications. It reviews the basic issues involved in the study of adaptive systems for signal processing. The virtues of this approach to statistical signal processing may be summarized as follows:
The use of an adaptive filtering algorithm, which enables the system to adjust its free parameters (in a supervised or unsupervised manner) in accordance with the underlying statistics of the environment in which the system operates. Hence, the need for determining the statistical characteristics of the environment is avoided.
Tracking capability, which permits the system to follow statistical variations (i.e. non-stationarity) of the environment.
The availability of many different adaptive filtering algorithms, both linear and non-linear, which can be used to deal with a wide variety of signal-processing applications in radar, sonar and biomedical imaging.
Digital implementation of the adaptive filtering algorithms, which can be carried out in hardware or software form.
In many cases these advanced algorithms trade robustness for improved performance. Furthermore, the improvements achieved are generally not uniform across all signal and noise environments of operational scenarios. The challenge is to develop a concept, which allows an appropriate mixture of these algorithms to be implemented in practical real time systems.
In general, improvements in system performance or array gain improvements are required when the noise environment of an operational system is non-isotropic, such as the noise environment of: (1) atmospheric noise or clutter (radar applications), (2) cluttered coastal waters and areas with high shipping density in which sonar systems operate (sonar applications), and the complexity of the human body (medical imaging applications). An alternative approach to improve the array gain of a real-time system requires the deployment of very large aperture arrays, which leads to technical and operational implications. Thus, the implementation of non-conventional signal processing schemes in operational systems will minimize very costly hardware requirements associated with array gain improvements.

15 KEYWORDS, DESCRIPTORS or IDENTIFIERS

(U) adaptive signal processing; synthetic aperture processing; signal cancellation; ultrasound imaging; interference; detection; estimation; sonar processing; radar processing; beamforming

#517567

CA020836